

CASAによるイメージング  
ビジビリティからイメージへ

# CASAの基本的な使い方

# CASAの起動

- # casapy <return>

起動メッセージとログウィンドウがあらわれる

```
[screen 0: casapy] — ssh — 84x32
-----> exit()
leaving casapy...
[0:orion]:ttsuka$ casapy           [~/observation/ALMA_SVdata/lecture/images]

=====
The start-up time of CASA may vary
depending on whether the shared libraries
are cached or not.
=====

CASA Version 4.2.1 (r29048)
Compiled on: Tue 2014/04/01 19:49:27 UTC

For help use the following commands:
tasklist          - Task list organized by category
taskhelp          - One line summary of available tasks
help taskname     - Full help for task
toolhelp          - One line summary of available tools
help par.parametername - Full help for parameter name

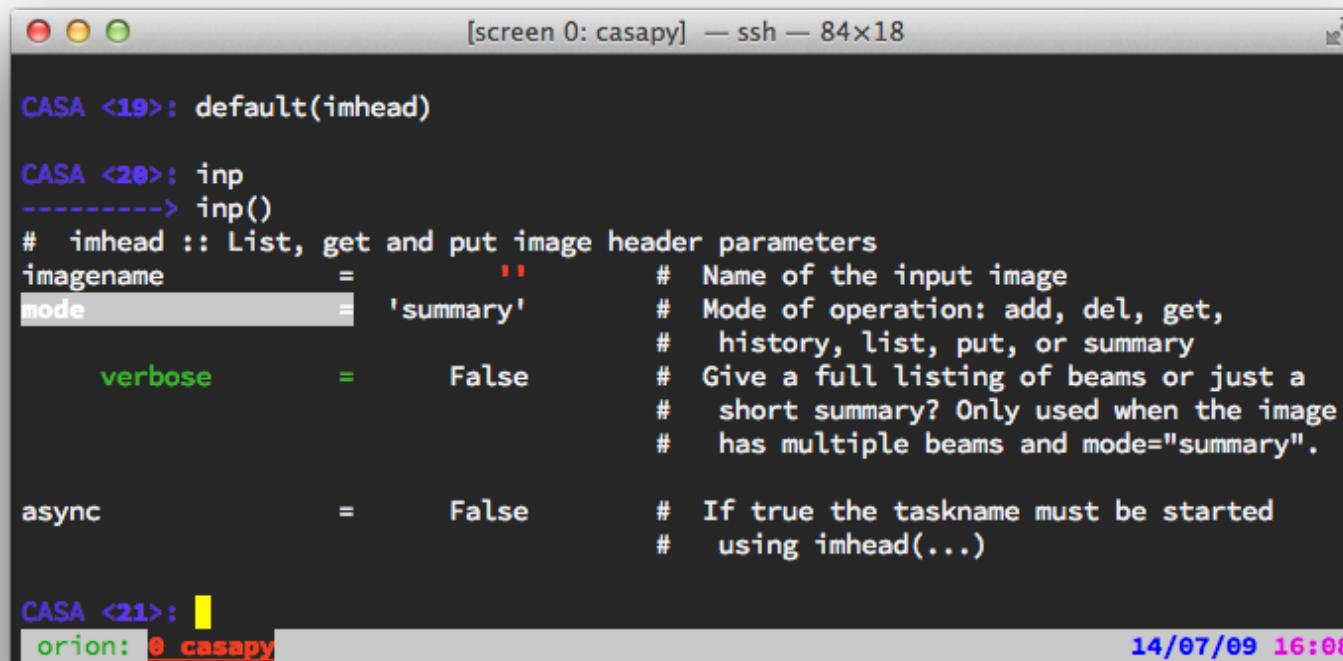
Activating auto-logging. Current session state plus future input saved.
Filename      : ipython-20140709-065942.log
Mode          : backup
Output logging : False
Raw input log  : False
Timestamping   : False
State          : active
*** Loading ATNF ASAP Package...
*** ... ASAP (4.2.0a rev#28807) import complete ***

CASA <2>: [ ]
orion: 0 casapy
```

Log Messages (orion:/home/ttsuka/observation/ALMA_SVdata/lecture/images/casapy-20140709-065937.log)			
Time	Priority	Origin	Message
...	INFO	...sa:::casa	---
...	INFO	...sa:::casa	CASA Version 4.2.1 (release r29047)
...	INFO	...sa:::casa	Tagged on: Thu, 27 Mar 2014

# タスク呼び出し/パラメータ入力/実行

- タスクの呼び出し
  - CASA<>: default([タスク名]) <return> ex) default(imhead) <return>
    - 初期値パラメータで呼び出される
  - CASA<>: tget [タスク名] <return>
    - 前回実行時のパラメータで呼び出される([タスク名].lastファイルが必要)
- タスクパラメータの確認
  - CASA<>: inp <return> or inp([タスク名]) <return>



The screenshot shows a terminal window titled "[screen 0: casapy] — ssh — 84x18". The window displays the following CASA session:

```
CASA <19>: default(imhead)

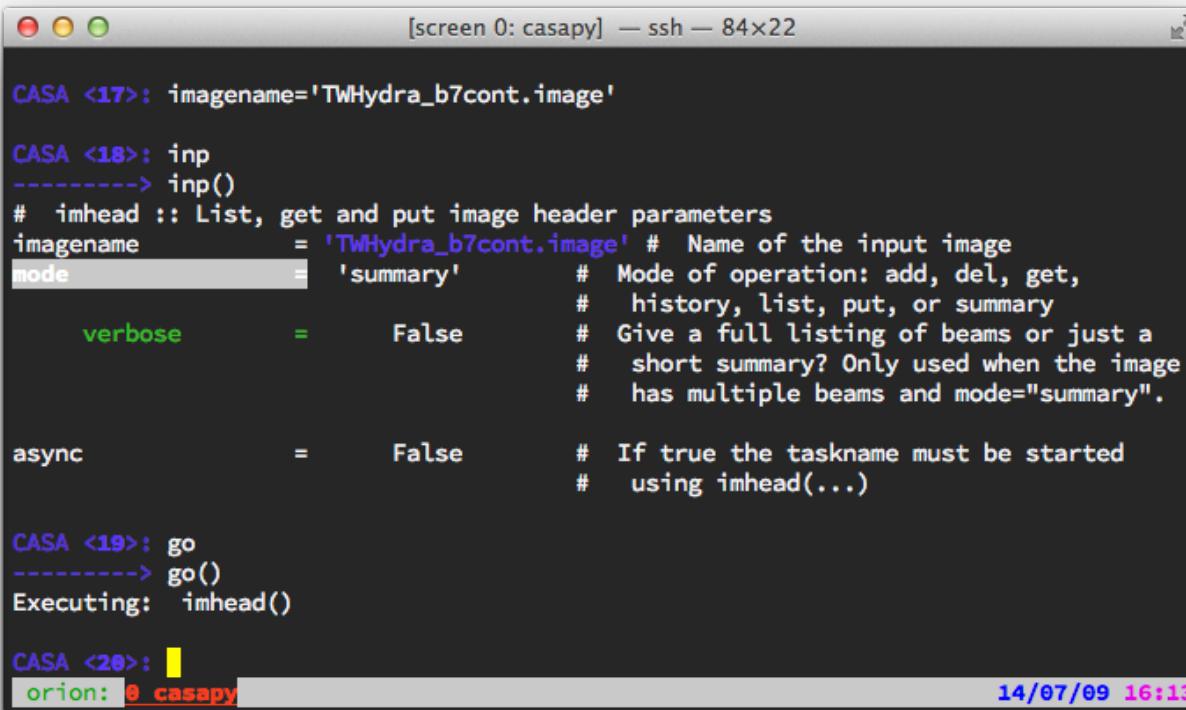
CASA <20>: inp
-----> inp()
# imhead :: List, get and put image header parameters
imagename      =      ""          # Name of the input image
mode           =  'summary'       # Mode of operation: add, del, get,
                                # history, list, put, or summary
verbose         =      False       # Give a full listing of beams or just a
                                # short summary? Only used when the image
                                # has multiple beams and mode="summary".
async          =      False       # If true the taskname must be started
                                # using imhead(...)

CASA <21>: [REDACTED]
orion: 0 casapy
```

The terminal window has a redacted area at the bottom where the user's input was typed. The status bar at the bottom right shows the date and time: "14/07/09 16:08".

# タスク呼び出し/パラメータ入力/実行

- パラメータの入力
  - CASA<>: [パラメータ名]= hogehoge <return>  
ex) imagename='TWHydra\_b7cont.image' <return>
- タスクの実行
  - CASA<>: go <return> / go([タスク名]) <return>
  - スクリプト的なタスクの実行も可能  
ex) imhead(imagename='TWHydra\_b7cont.image') <return>



The screenshot shows a terminal window titled "[screen 0: casapy] — ssh — 84x22". The window displays a CASA script being run. The script starts with setting the image name, followed by defining parameters for the imhead task (mode, verbose, async), and finally executing the task with go(). The status bar at the bottom shows "orion: 0 casapy" and the date/time "14/07/09 16:13".

```
CASA <17>: imagename='TWHydra_b7cont.image'

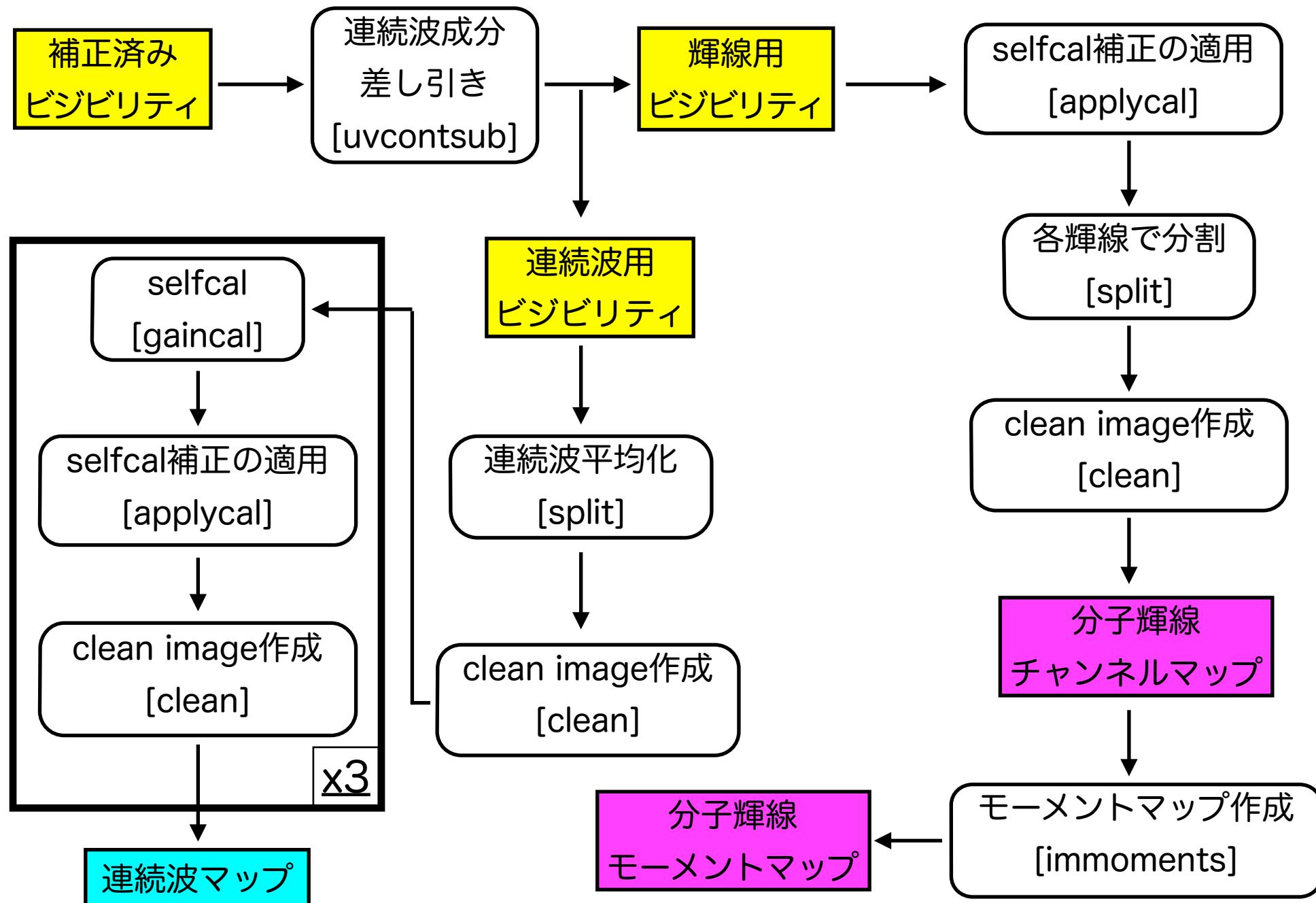
CASA <18>: inp
-----> inp()
# imhead :: List, get and put image header parameters
imagename      = 'TWHydra_b7cont.image' # Name of the input image
mode           = 'summary'          # Mode of operation: add, del, get,
#                   history, list, put, or summary
verbose        = False             # Give a full listing of beams or just a
#                   short summary? Only used when the image
#                   has multiple beams and mode="summary".

async          = False             # If true the taskname must be started
#                   using imhead(...)

CASA <19>: go
-----> go()
Executing: imhead()

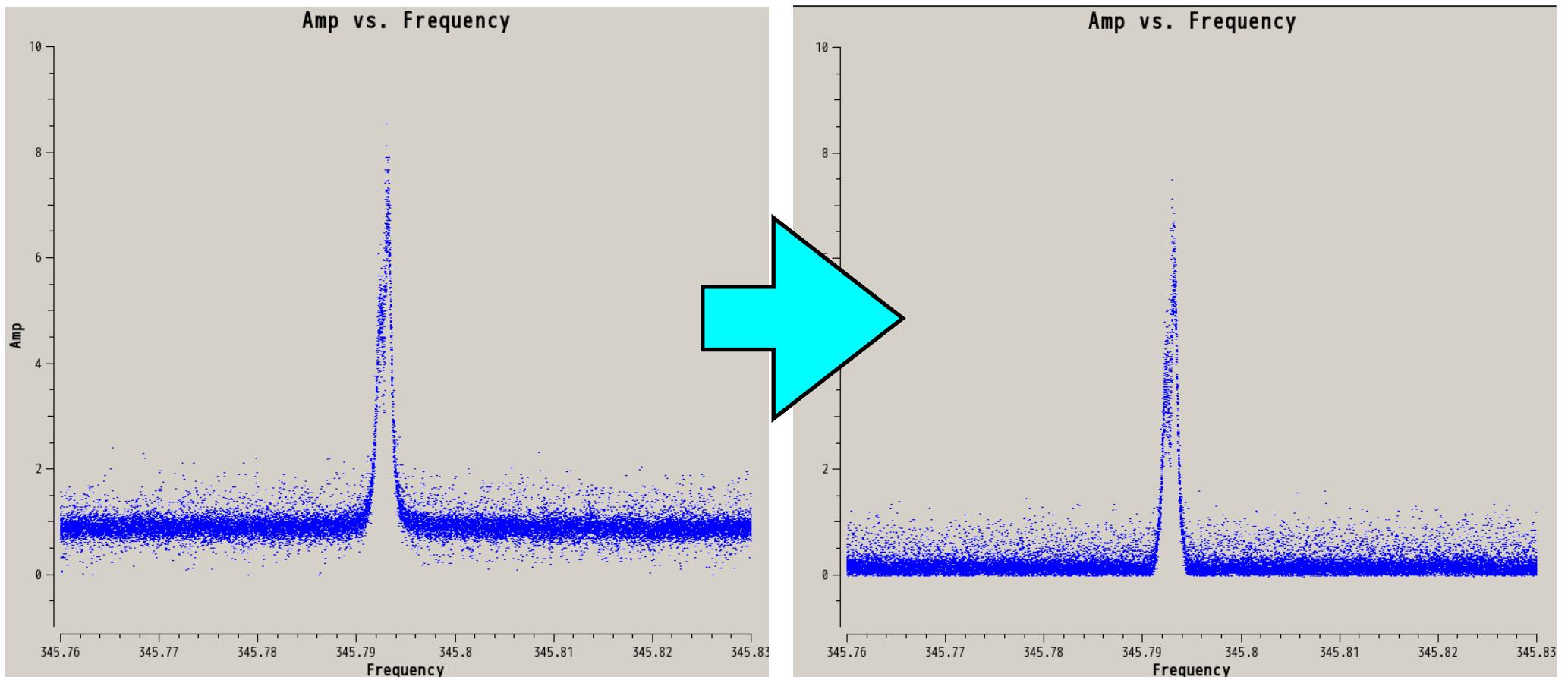
CASA <20>: [redacted]
orion: 0 casapy
```

# リダクションチャート



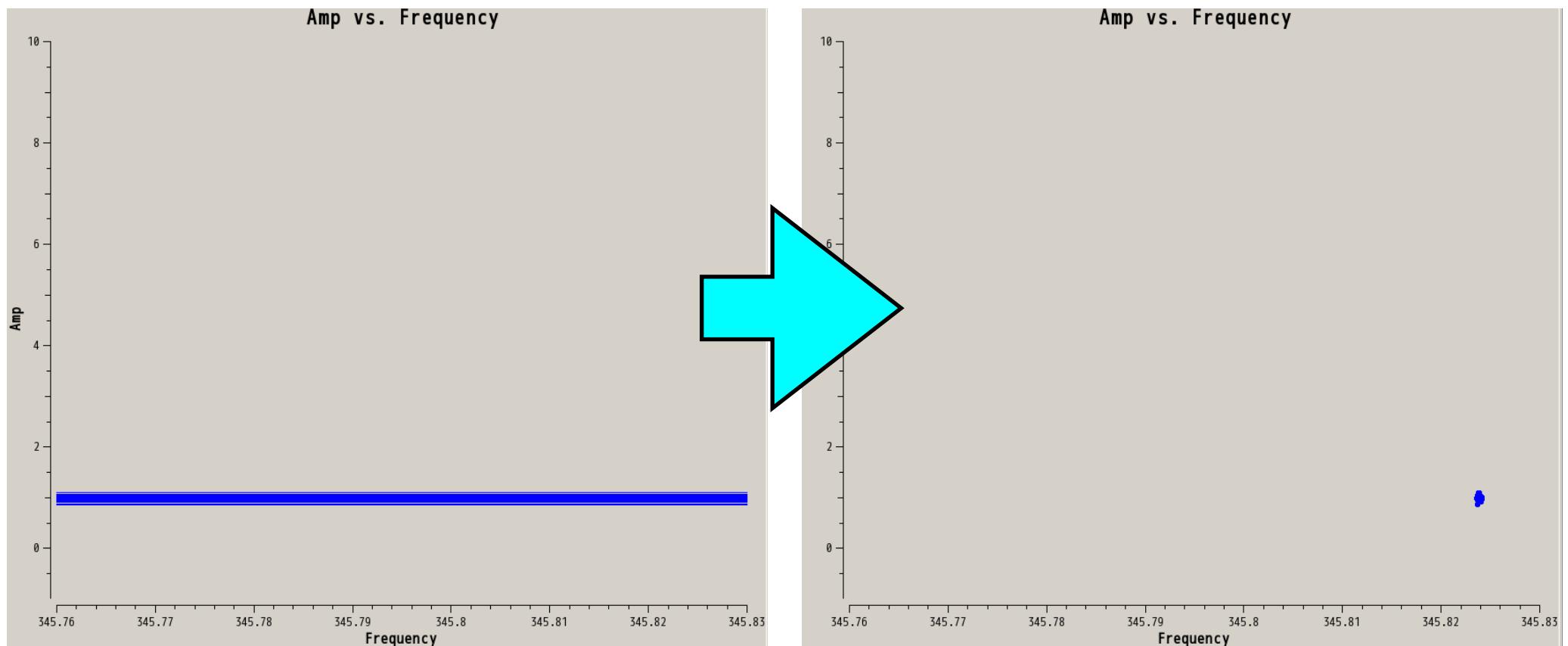
# 連続波成分の差し引き

- CASAタスク: uvcontsub
  - ビジビリティデータ内で差し引く
  - fitspw: ベースラインとなる範囲をチャンネルで指定する
  - combine: 連続波成分を計算する際にspwやscanを足して計算するかどうか
  - want\_cont: 連続波成分もファイルとして保存する



# 連続波ビジビリティの平均化

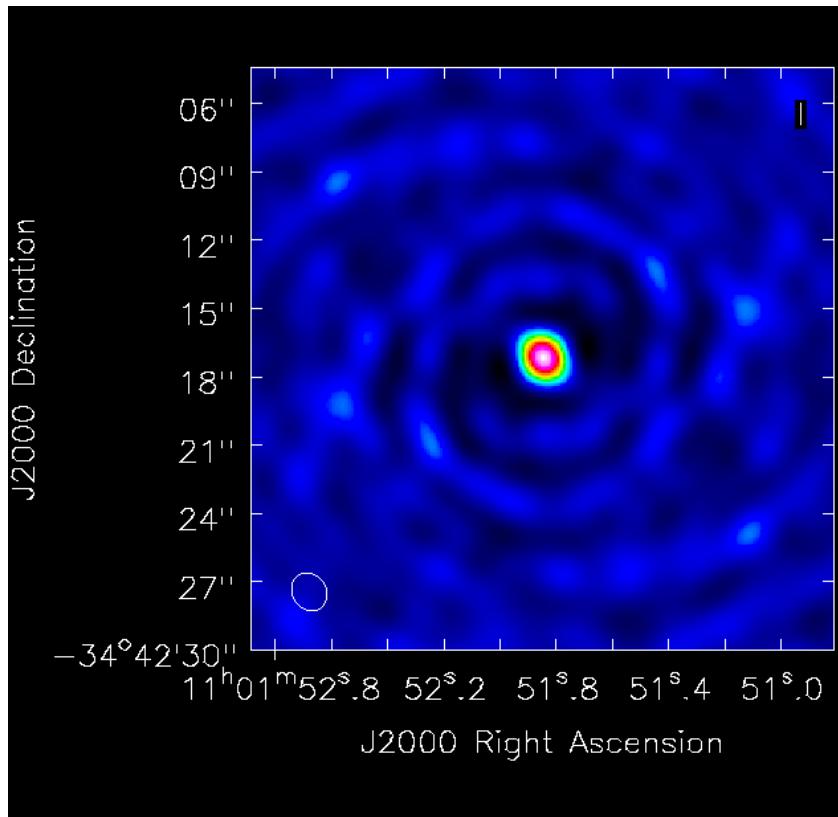
- uvcontsubで吐き出された連続波データはデータ量が膨大。今後の連続波イメージング+セルフキャリブレーション時に時間がかかるてしまう
- なのであらかじめ連続波データを平均+チャネル数減少させておく



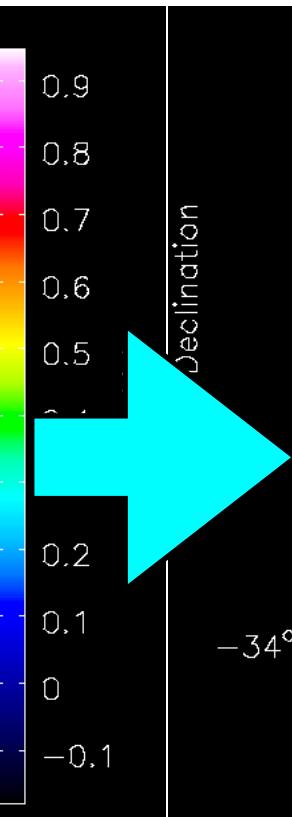
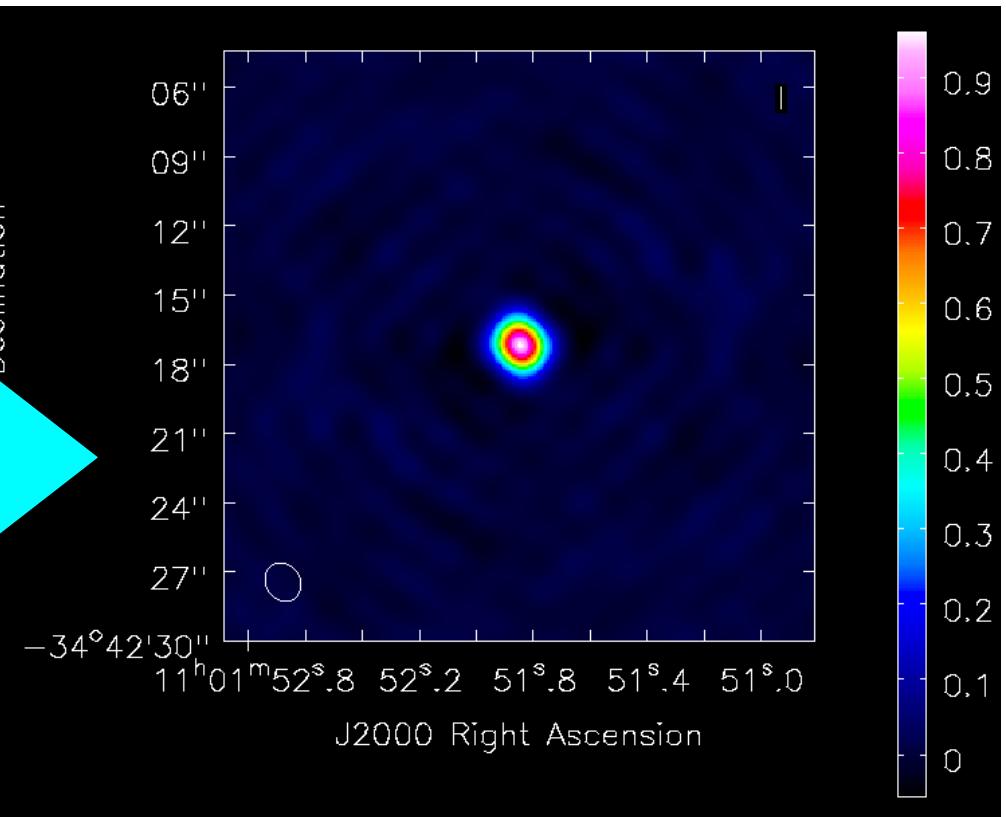
# CLEAN

- CASA task: clean  
ビジビリティのFFT + イメージのdeconvolution(CLEAN)

dirty map



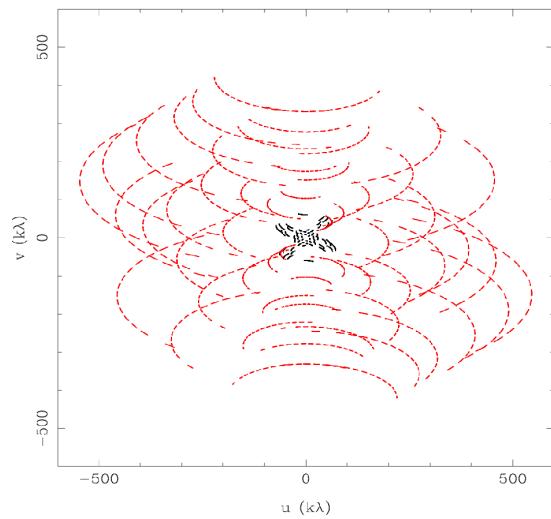
clean map



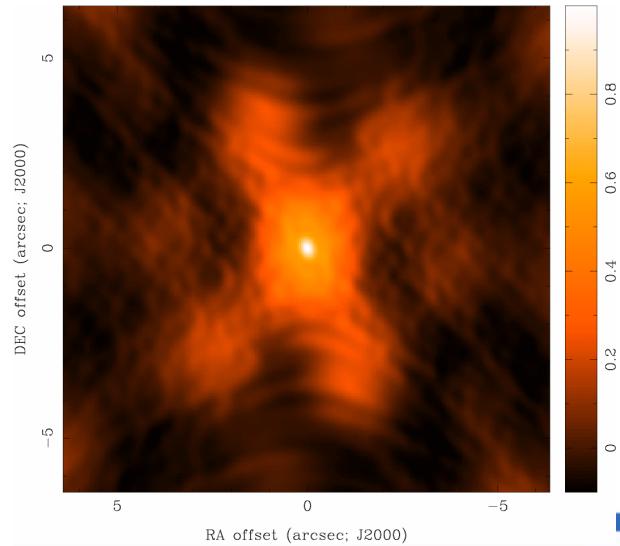
# CLEANのパラメータ説明(抜粋)

- mode: チャンネル方向データの取り扱い方。連続波データでは'mfs'、輝線データでは'velocity'や'frequency'
- imsize: イメージ正方格子の縦横のピクセル数
  - 観測視野より大きくする(~倍くらい)
- cellsize: 1ピクセル辺りの角度
  - 観測分解能より十分に小さくする(~1/10くらい)
- niter: CLEANを行う(ピークを走査する)全回数
  - 回数終了でCLEAN停止
- threshold: CLEANを停止させる値
  - niter数が残っていてもCLEANが終わる
- weight: ビジビリティを逆フーリエ変換する際のウェイトを決める
  - natural weight: 分解能は悪いがノイズレベルは低い。(UV平面の粗密でウェイト)
  - uniform weight: ノイズレベルが高くなるが分解能を最大限生かす。(UV平面で一定)
  - briggs weight: 上記二つの中間。robust parameterで調整。(-2でuniform, 2でnatural)
- uvtaper: uvdistanceに対するガウシアン関数でウェイトをかける

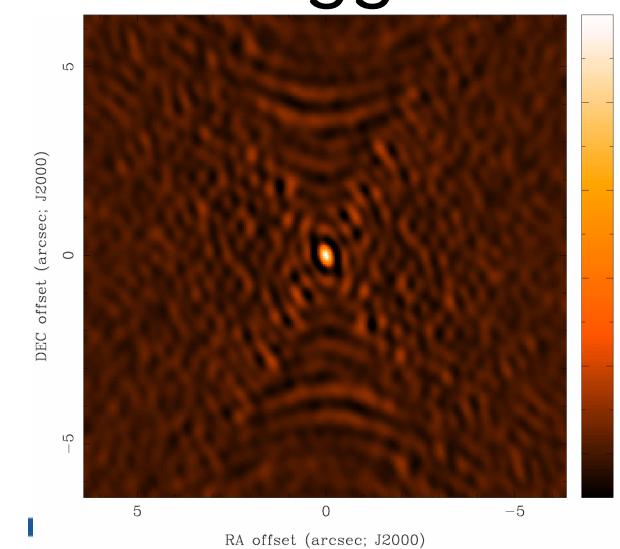
# ウェイトと合成ビームの例



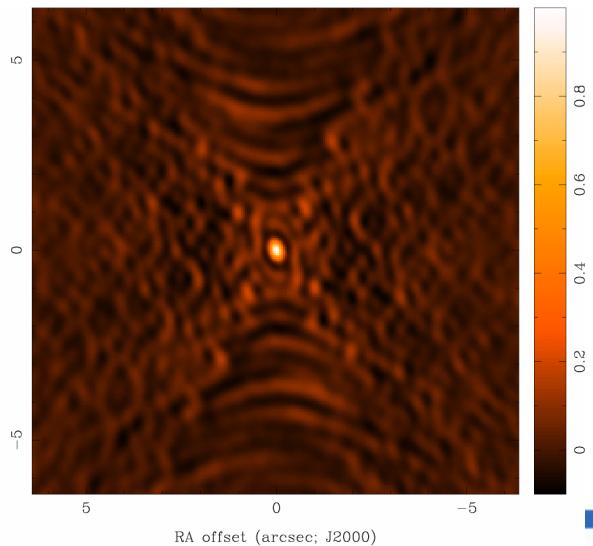
Natural



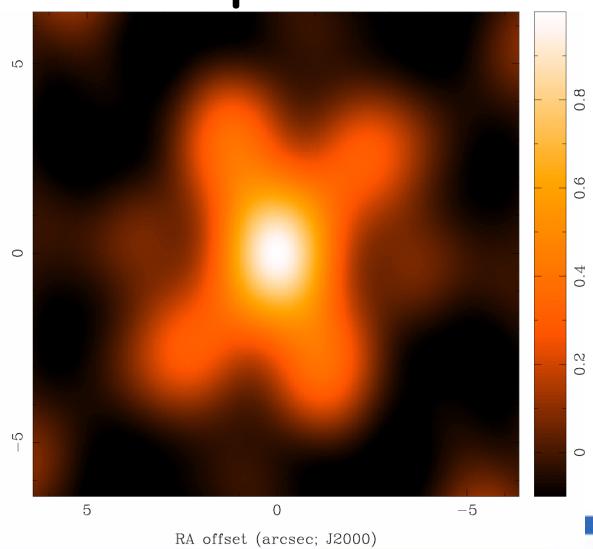
Briggs



Uniform



Tapered

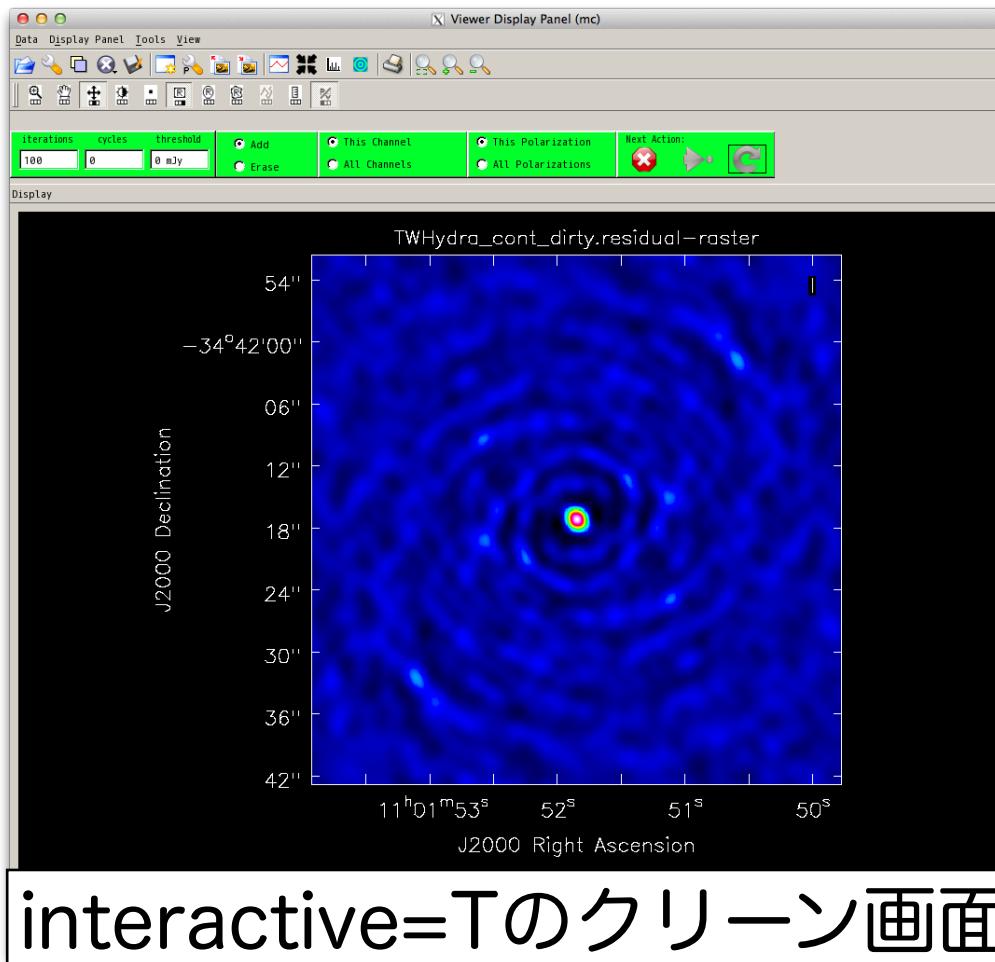


# Clean : 注意点

- イメージの上書きが出来ない
  - `imagename='TWHydra_CO3_2line'`
- 一度クリーンを行い、再度クリーンを行う場合
  - `CASA> os.system('rm -rf TWHydra_CO3_2line.*')`が必要

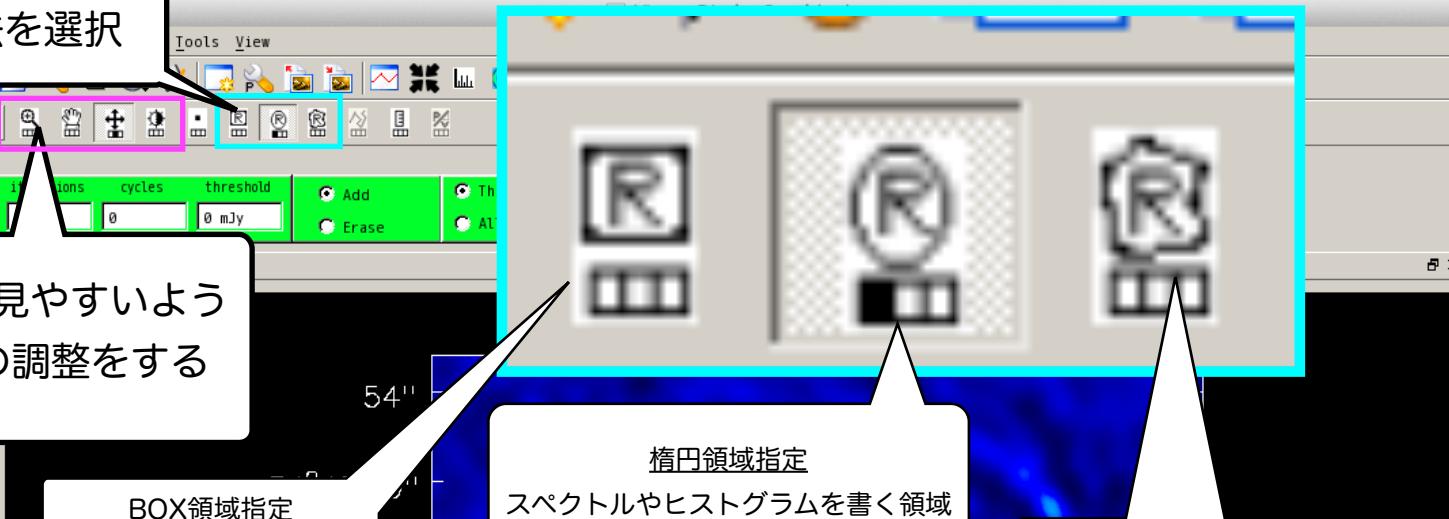
# Interactive CLEAN

- CLEANで用いるマスクは、実際の放射(残差マップ)を見ながらインタラクティブに調整する事が可能
- interactive=Trueにする
- go(clean)で実行するとGUIが立ち上がる



# Interactive CLEAN: 領域の指定

2,領域指定の手法を選択

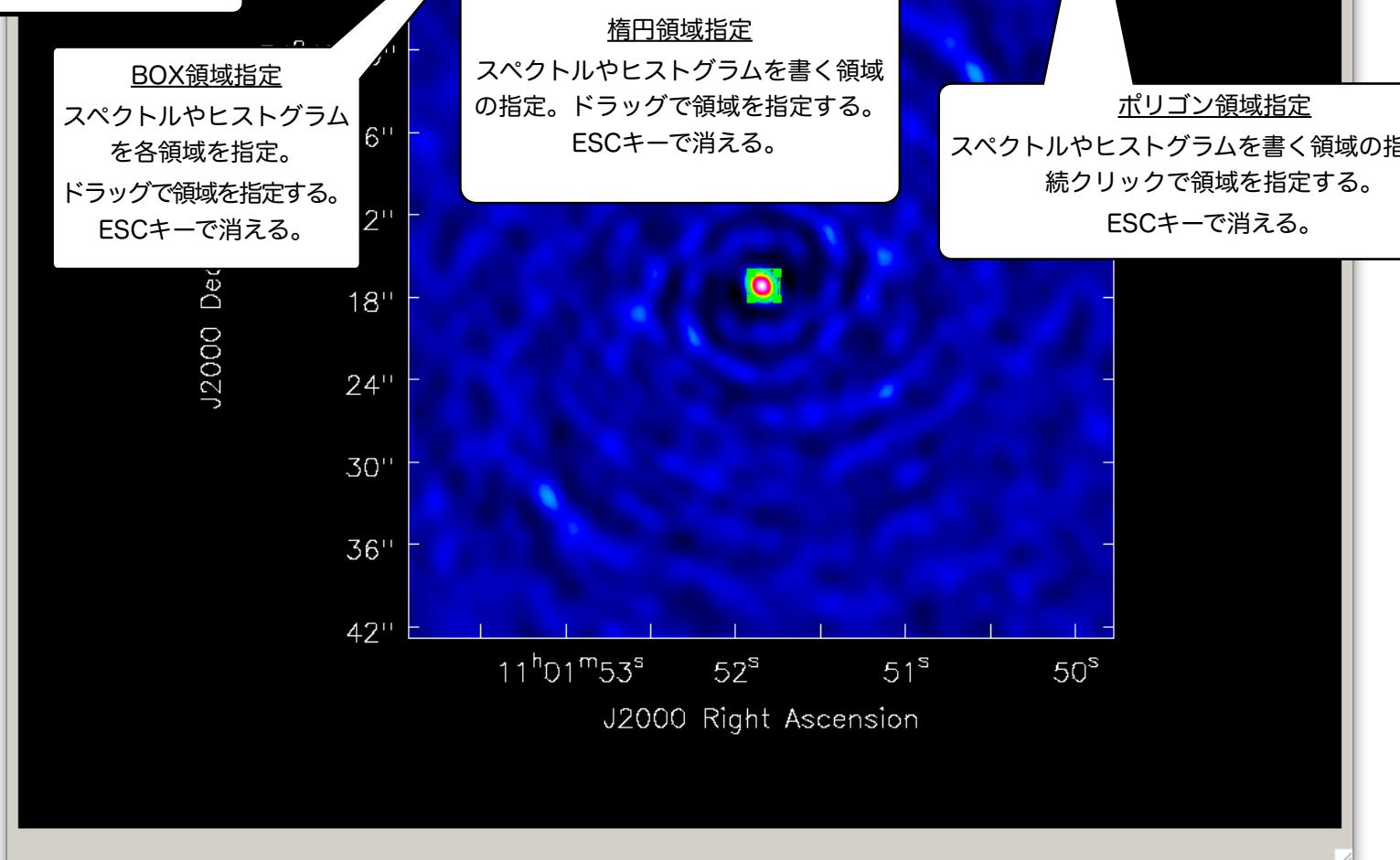


1,指定したい領域が見やすいように、拡大やカラーの調整をする

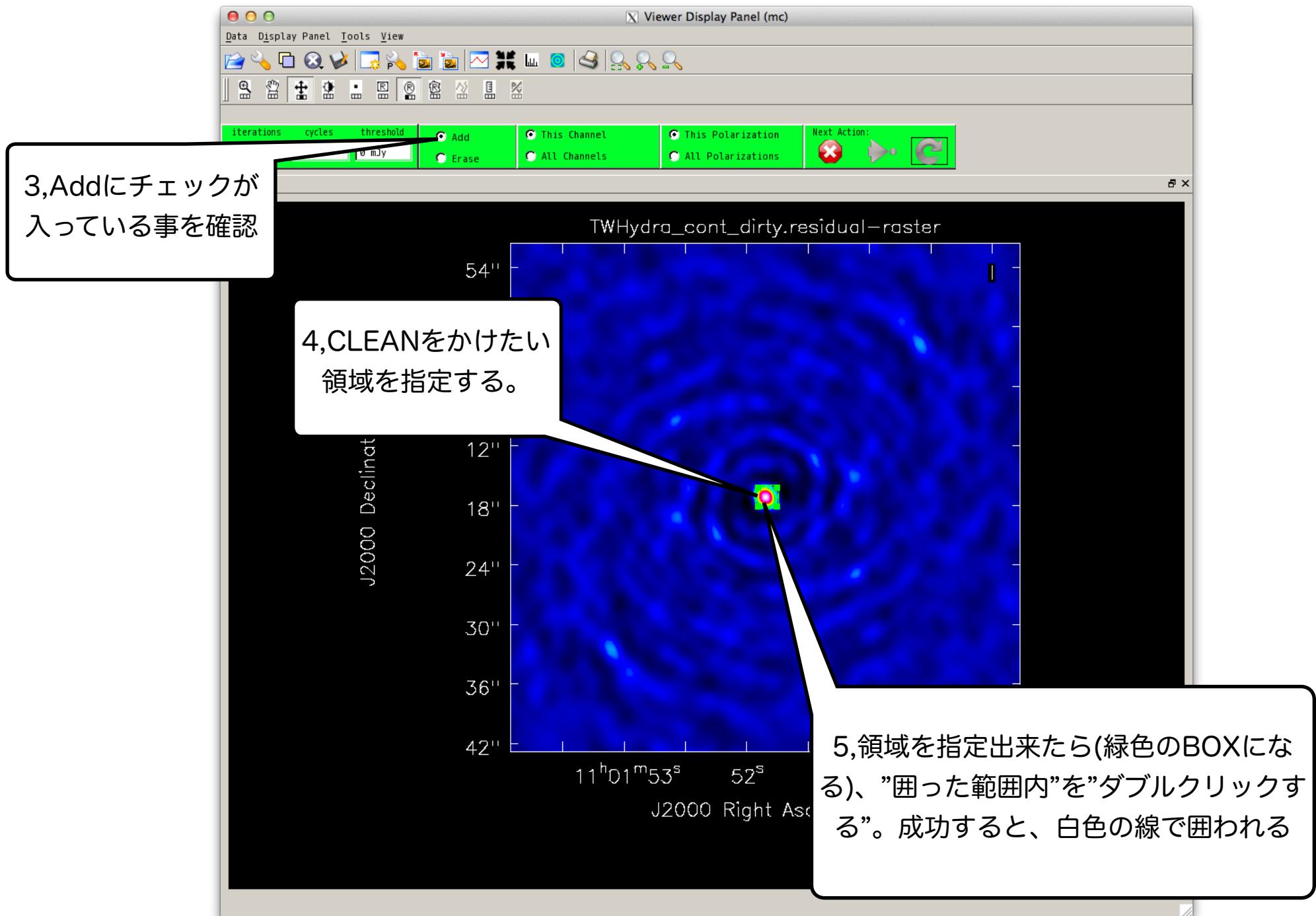
BOX領域指定  
スペクトルやヒストグラム  
を各領域を指定。  
ドラッグで領域を指定する。  
ESCキーで消える。

楕円領域指定  
スペクトルやヒストグラムを書く領域  
の指定。ドラッグで領域を指定する。  
ESCキーで消える。

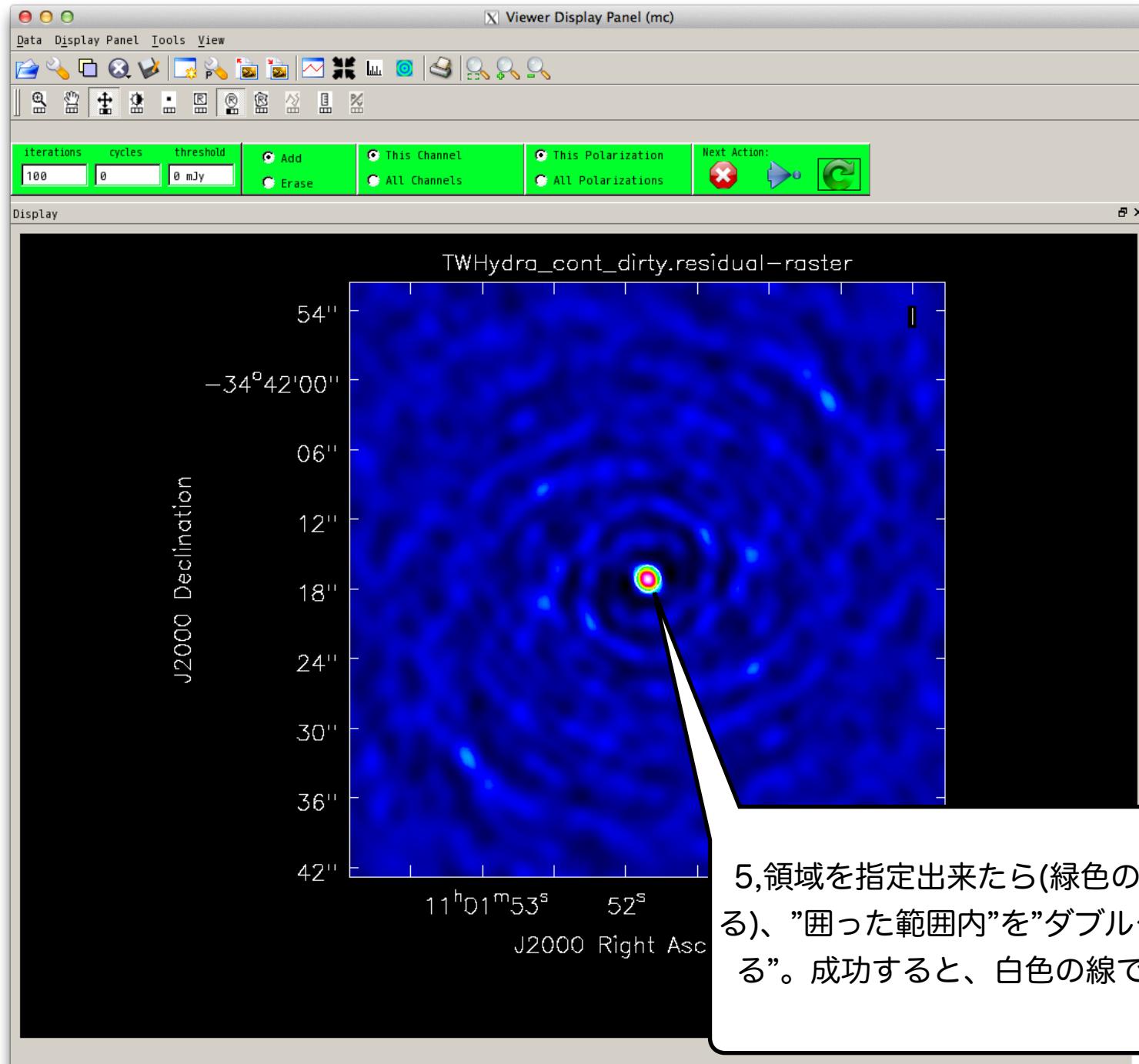
ポリゴン領域指定  
スペクトルやヒストグラムを書く領域の指定。連  
続クリックで領域を指定する。  
ESCキーで消える。



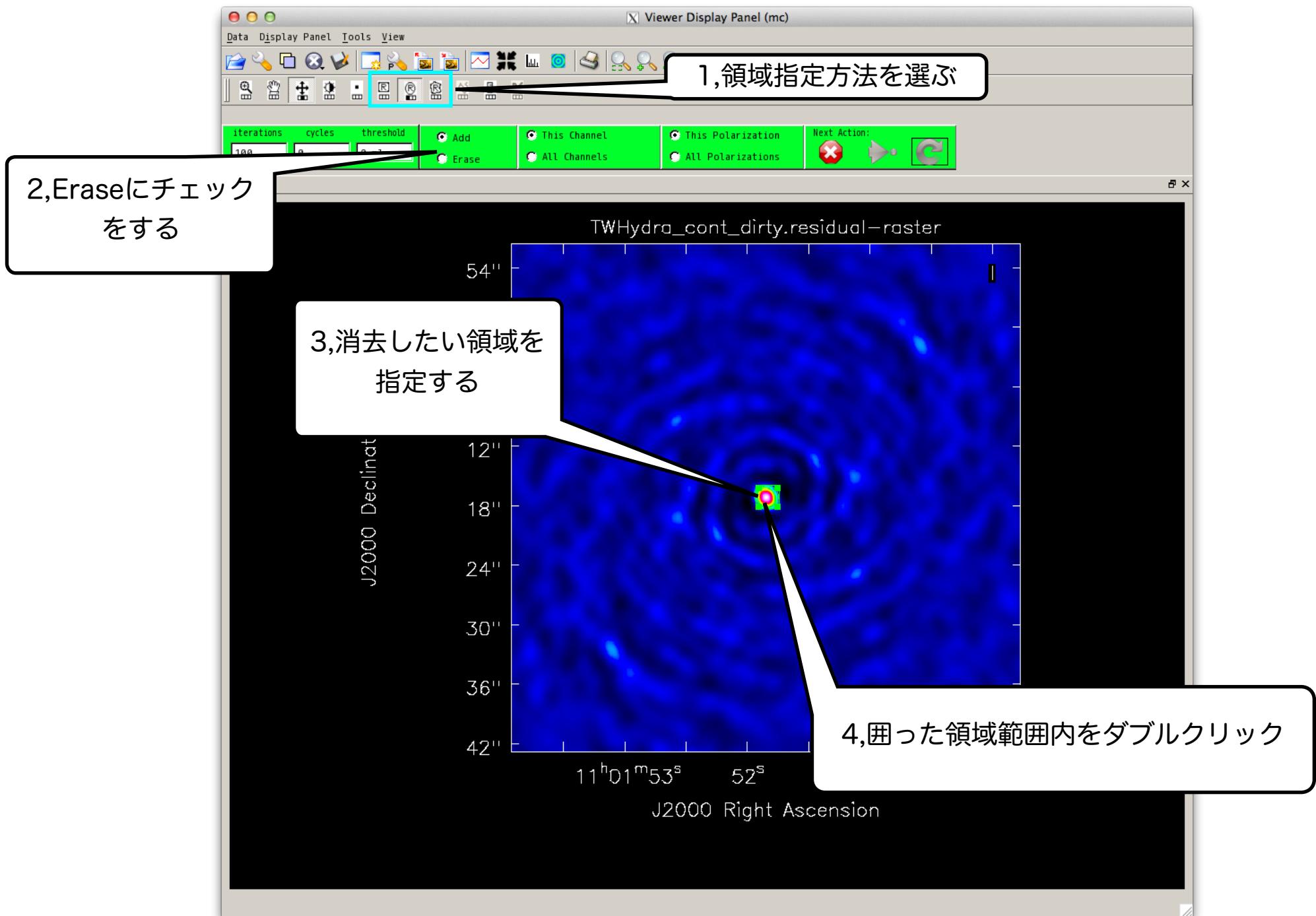
# Interactive CLEAN: 領域の指定



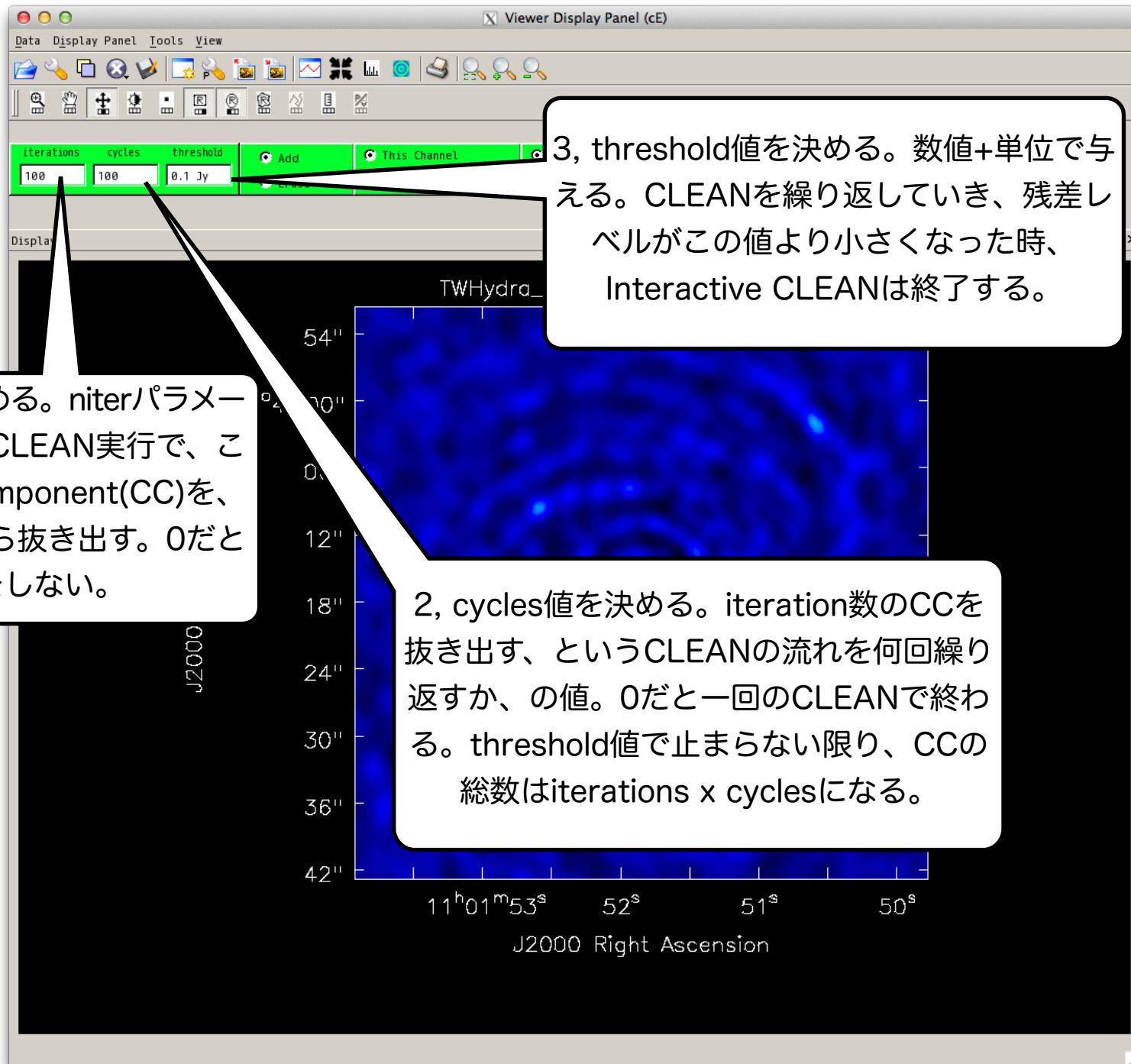
# Interactive CLEAN: 領域の指定



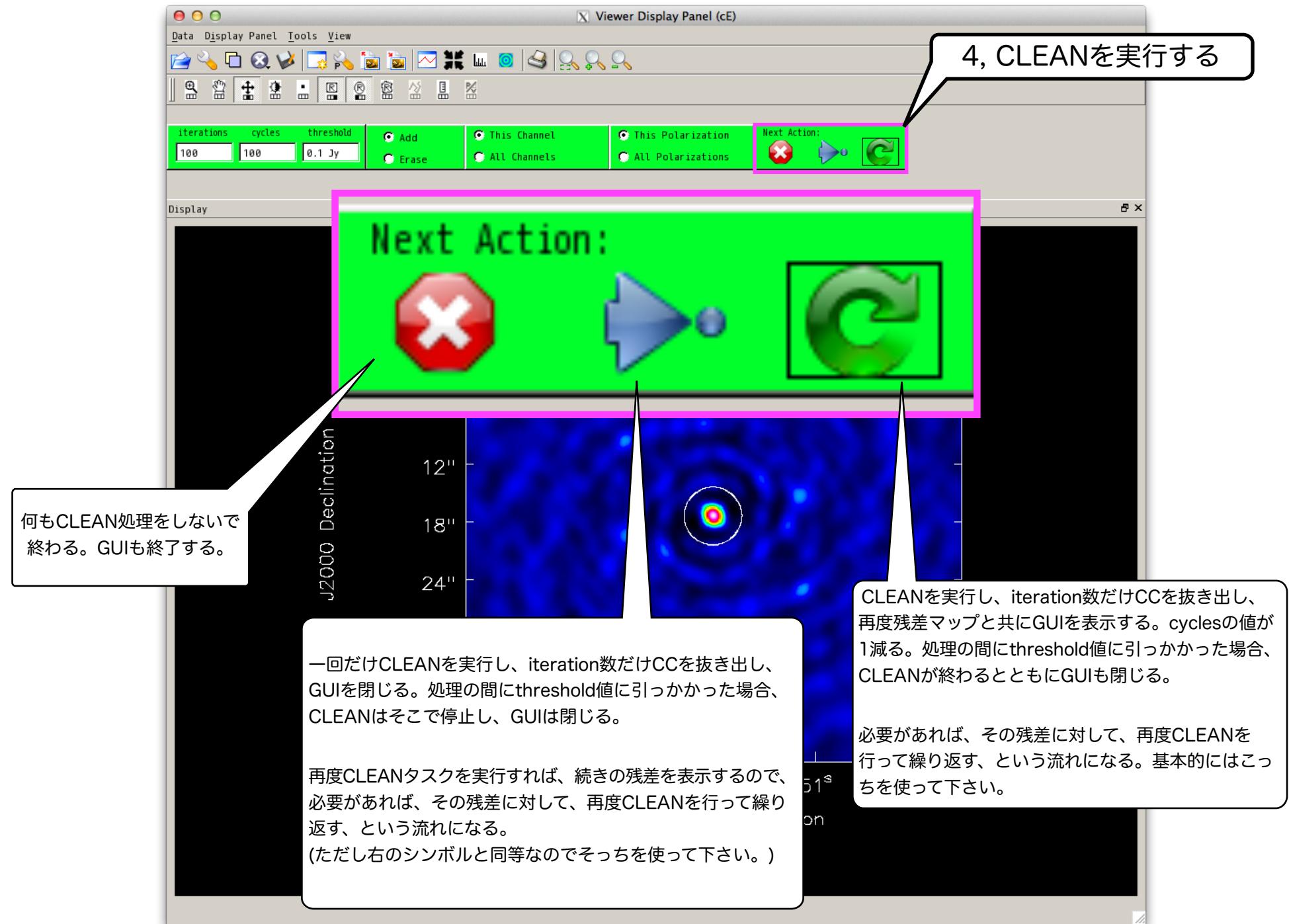
# Interactive CLEAN: 領域の消去



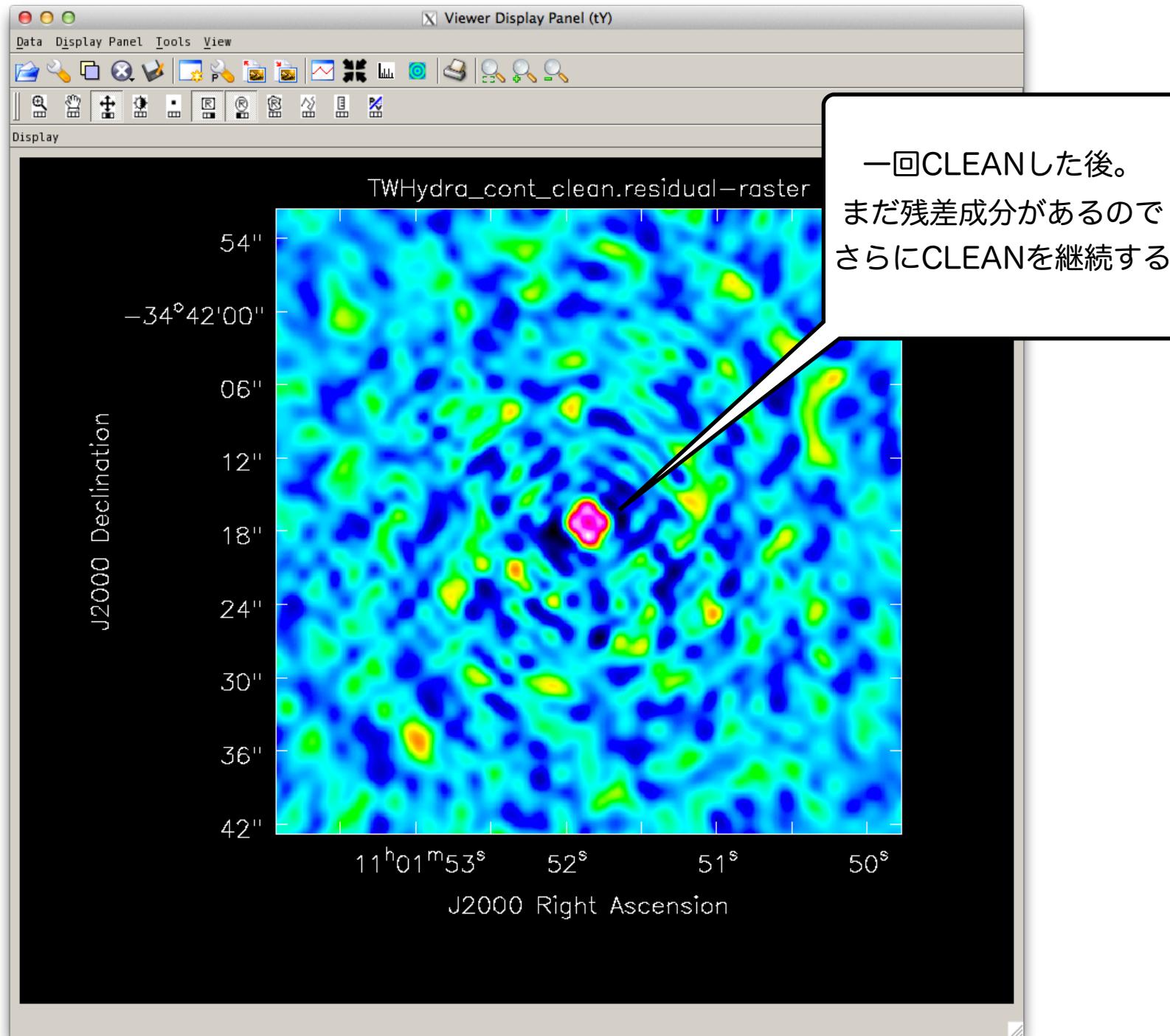
# Interactive CLEAN: CLEANを行う



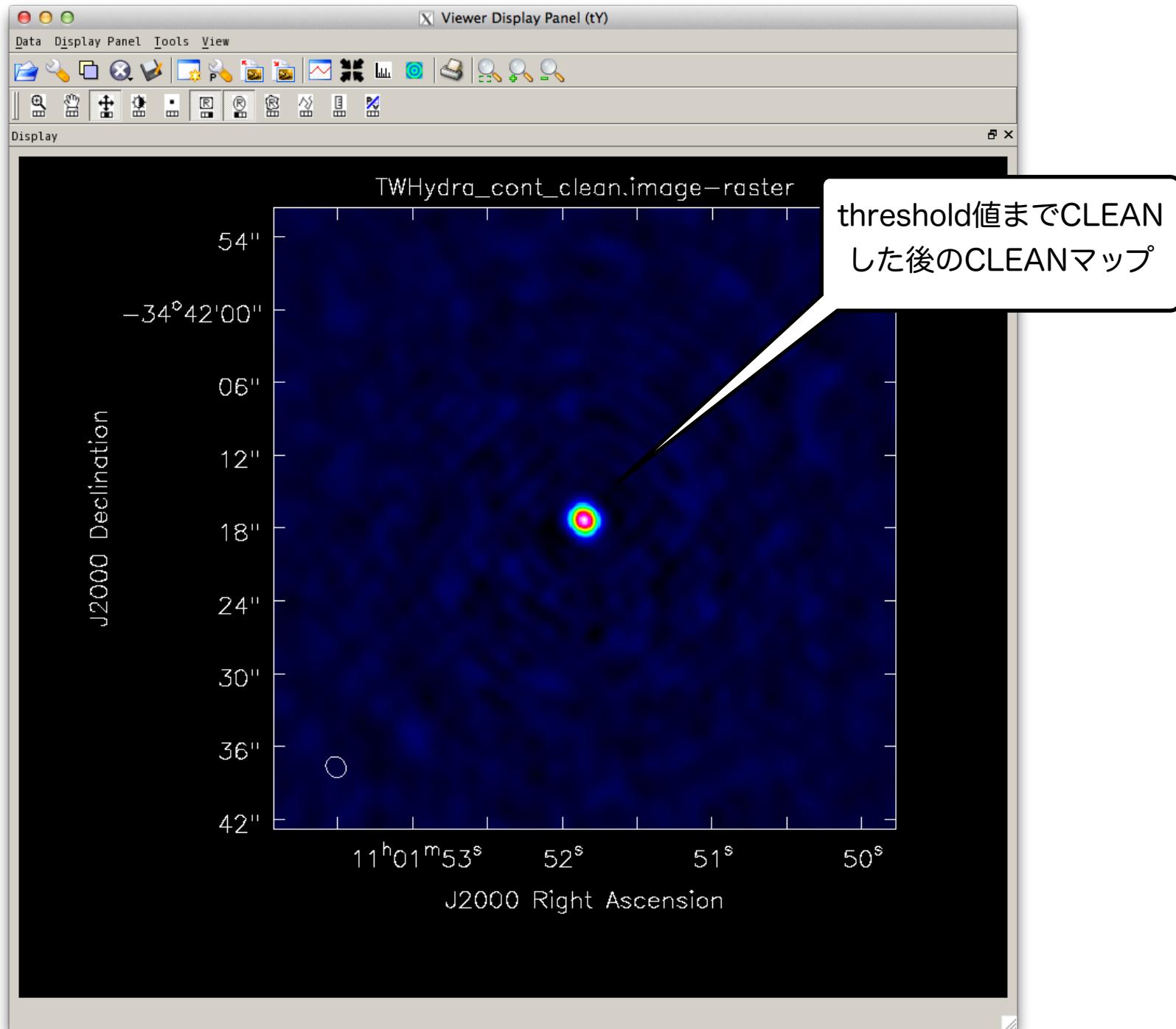
# Interactive CLEAN: CLEANを行う



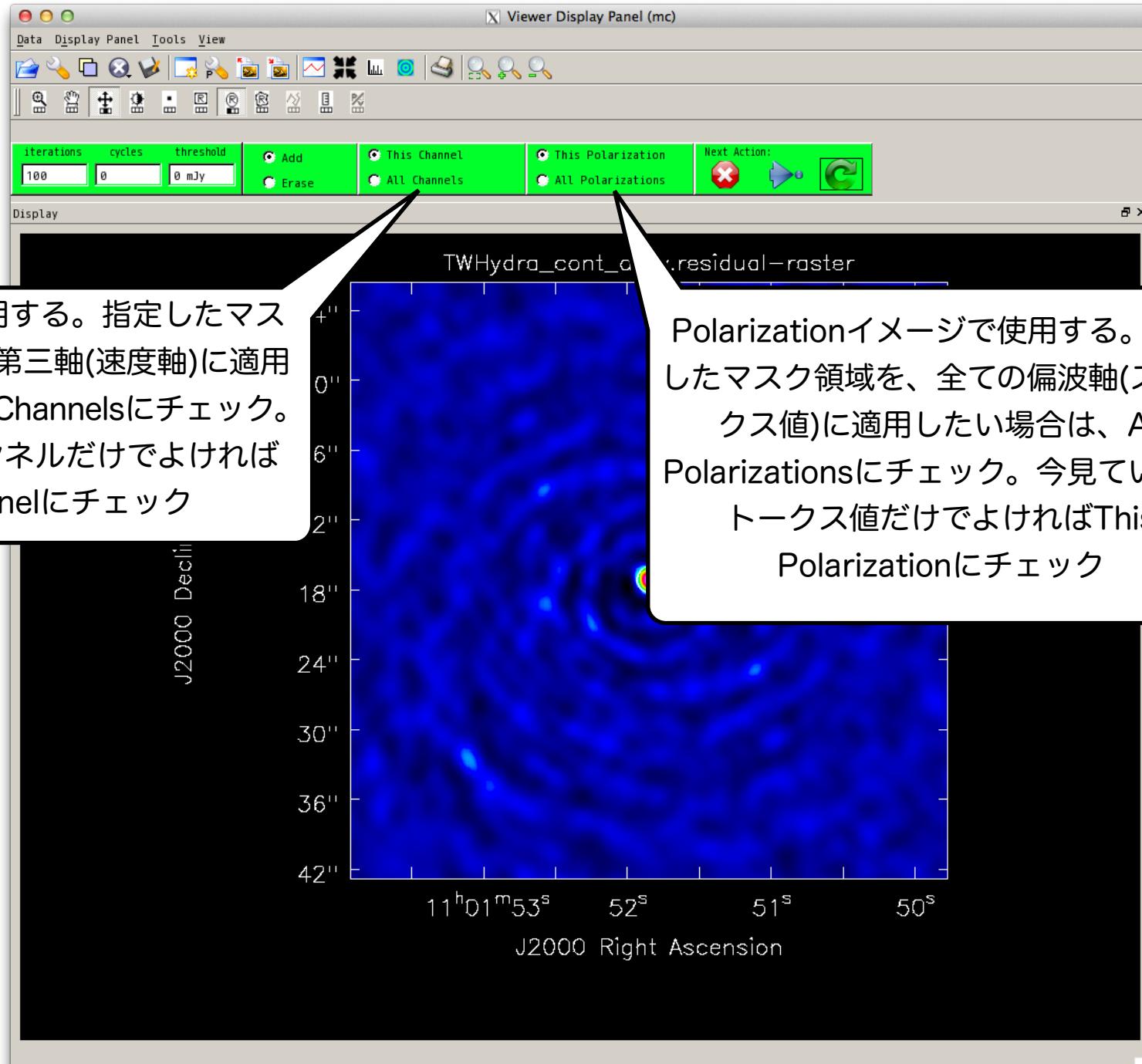
# Interactive CLEAN: CLEANを行う



# Interactive CLEAN: CLEANを行う



# Interactive CLEAN: その他のパラメータ



# self-calibration: 基本方法

- モデルに合うようにビジビリティの複素ゲイン項を解く
  - 位相を動かすので、絶対位置情報が失われる
  - 間違ったモデルを入力してしまうと間違った解になる
    - シンプルかつSNの良いデータを使うのが望ましい
    - 今回のモデルでは初回のCLEANのイメージをモデルとして用いる
      - 元ビジビリティデータ(\*.ms)のmodel columnにインプットされている

1.ゆっくりした時間スケールでの位相補正から開始

(1st - phase only selfcal)

2.徐々に時間スケールを短く (2nd - phase only selfcal)

- 変動の激しい位相項を先に解いてしまう

3.最後に振幅項も含めて解く (final - amp&phase cal)

- 位相が収束していないと変な振幅になる場合がある

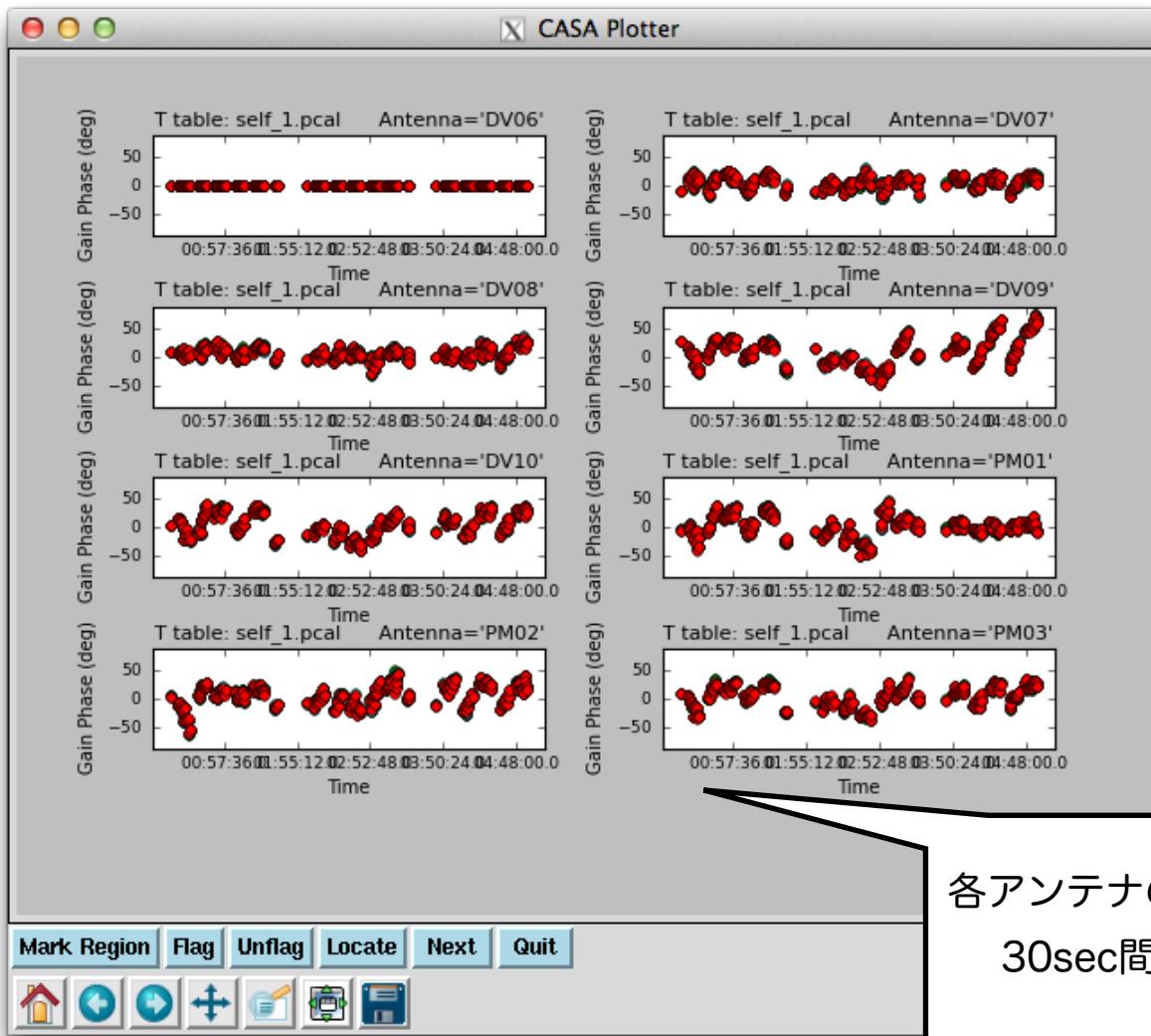
# self-calibration: 1st round

- CASA task: gaincal
  - caltable: 出力する補正項ファイル(ゲインテーブル)の名前
  - gaintype: 複素ゲインを計算する際の偏波データの取り扱い  
‘G’-偏波毎に計算、’T’-偏波を平均して計算
  - calmode: 計算項の指定。‘p’(phase), ‘a’(amp), ‘ap’(amp&phase)
  - solint: 計算する時間間隔の指定
  - refant: 位相参照となるアンテナ
- 最初は位相項のみを大きい時間スケールで解く
  - calmode=‘p’
  - solint=‘30s’ - 観測の積分間隔が10sなので、 $\sqrt{3}$ 倍感度が良くなる
  - 徐々に時間を細かくし、最終的に観測の積分間隔にする
- 偏波を平均して一つの解とする
  - gaintype=’T’ -  $\sqrt{2}$ 倍感度が良くなる

# self-calibration: 1st round

- CASA task: plotcal

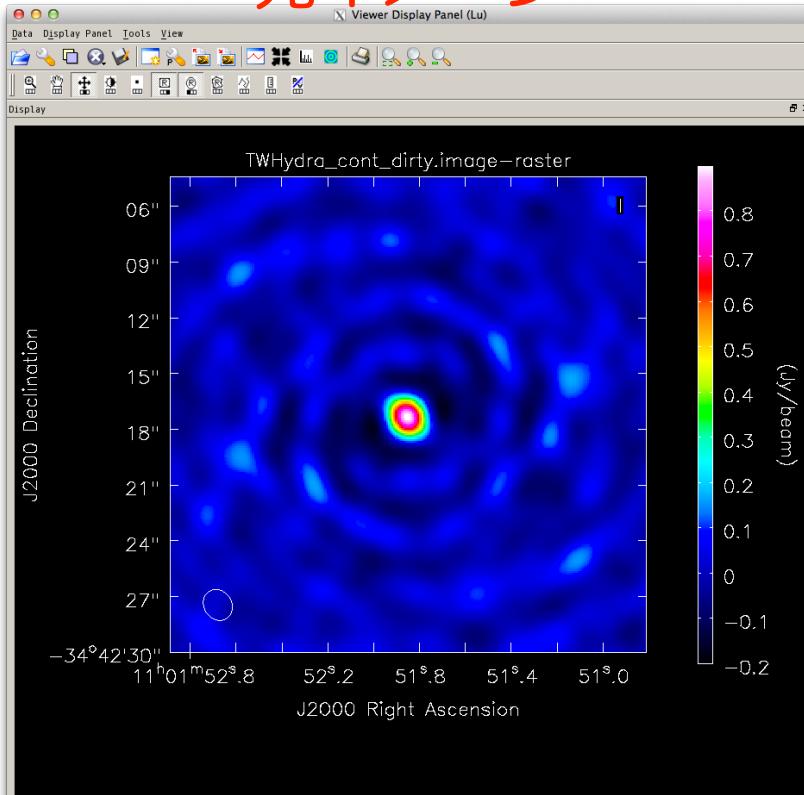
- gaincalで作成したゲインテーブルをプロットする
- selfcalの場合はxaxis='time'で確認する



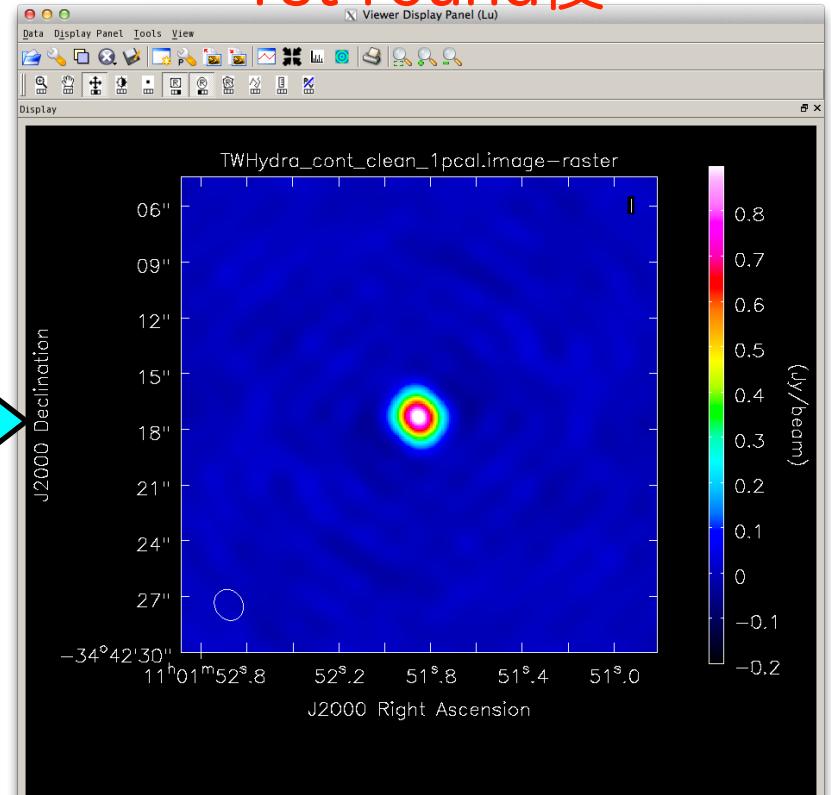
# self-calibration: 1st round

- CASA task: applycal
  - 作成したゲインテーブルを元ビジビリティデータに適用する
  - 補正したビジビリティデータは、.msファイルのcorrected columnに入る
- 補正後のビジビリティを用いて再度CLEANを行う。それを元に self-calibrationの2nd roundへ。

元イメージ

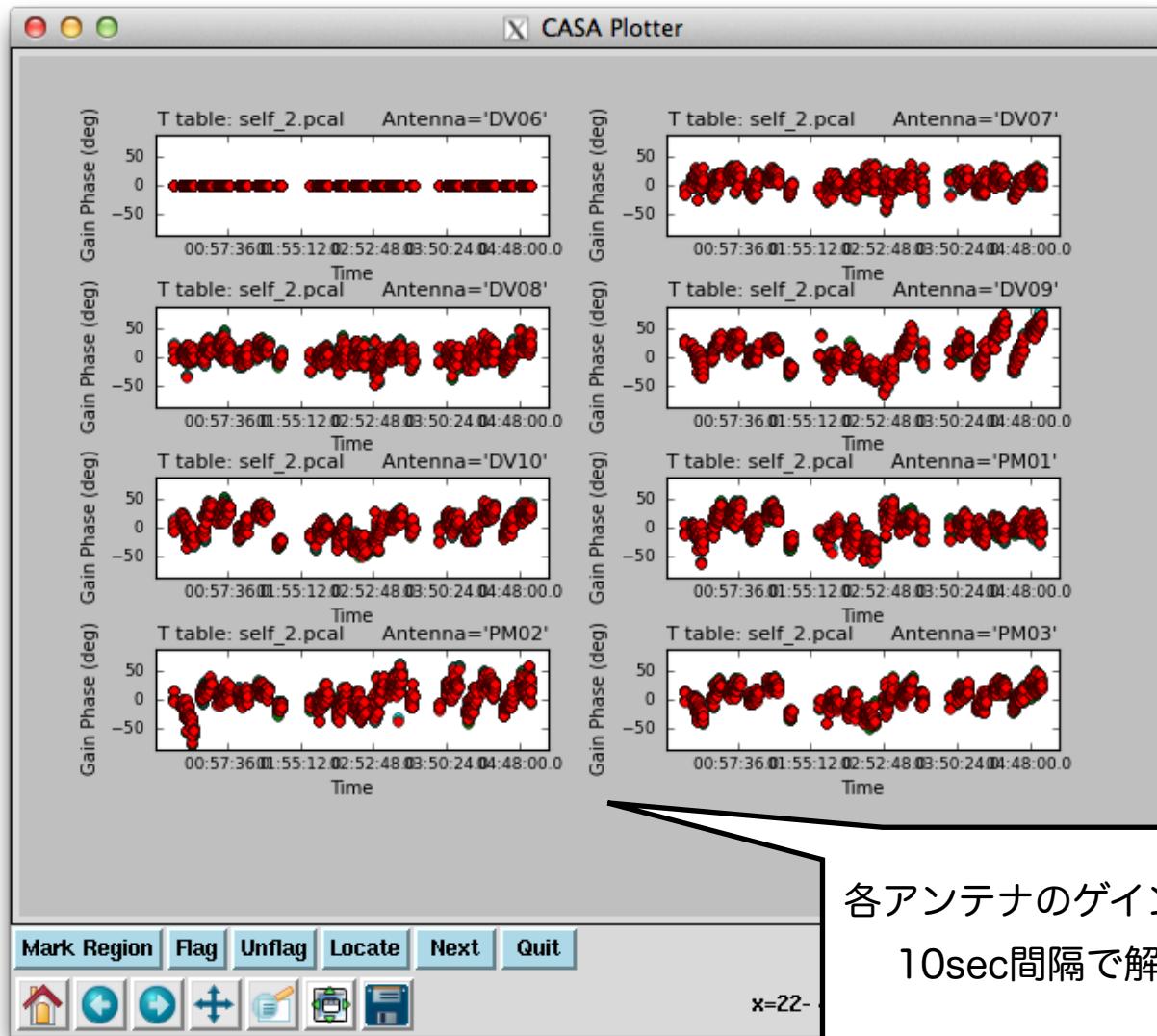


1st round後



# self-calibration: 2nd round

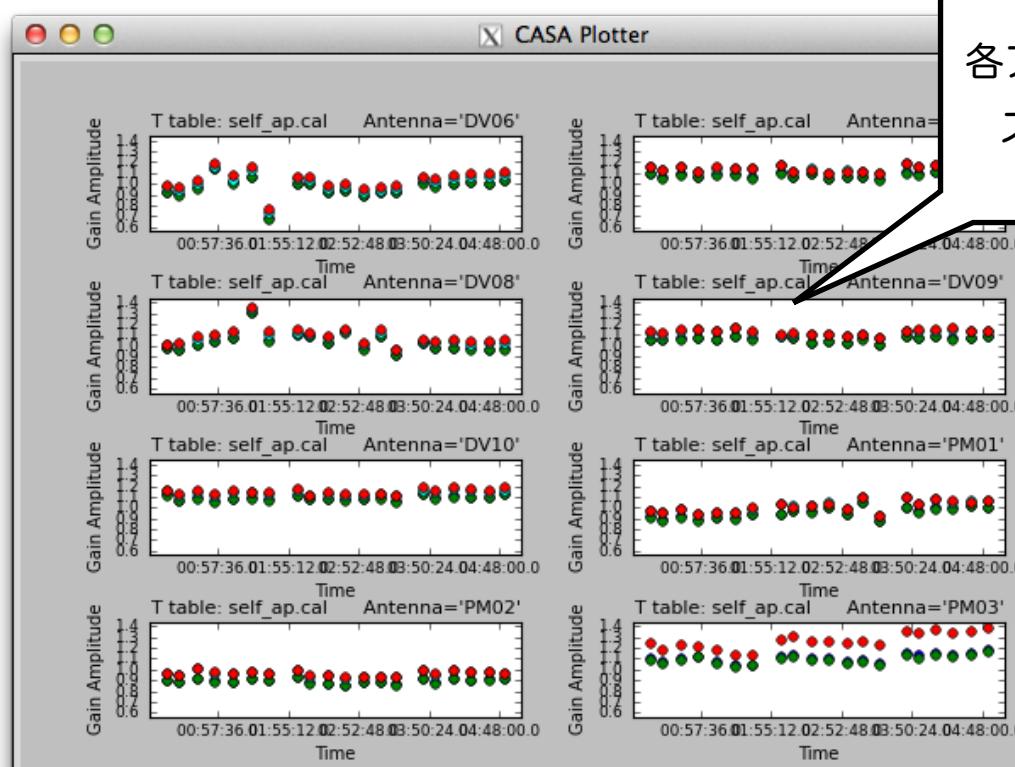
- 時間スケール細かくして解く
  - solint='int' - 観測の積分間隔と同値(10s)
  - 細かくする間隔を小分けにすればより慎重なself-calibrationとなる



各アンテナのゲインテーブルのプロット  
10sec間隔で解があることを確認

# self-calibration: amp.&phase

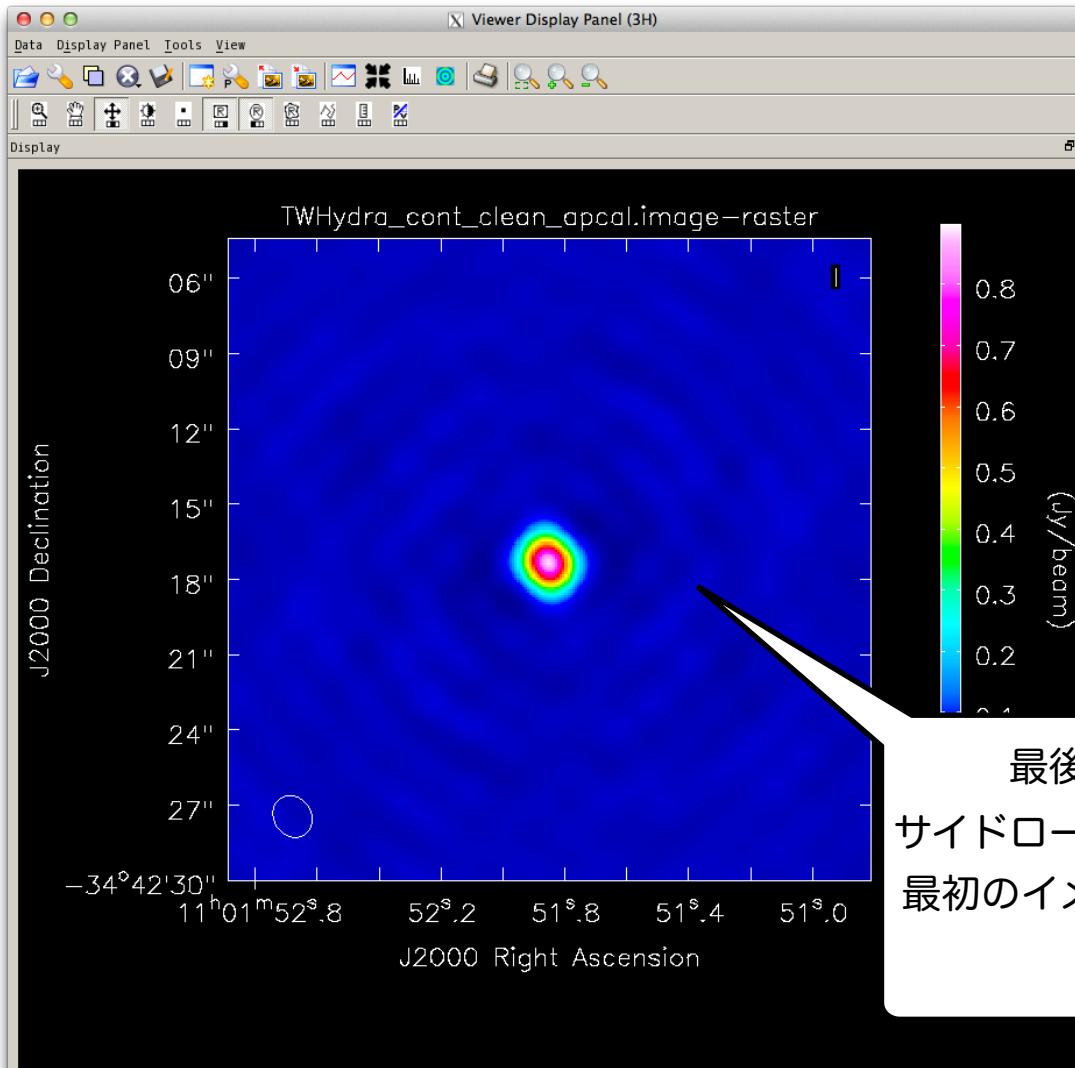
- 解いた位相補正テーブルを適用しつつ、振幅項を解く
  - gaintable=['self\_2.pcal'] # 位相補正テーブル名
  - calmode='ap'
- 振幅の変化は位相に比べて遅いので、計算間隔はスキャン単位で
  - solint='inf' : infinite, up to boundaries controlled by combine
  - combine='': solution per scan



各アンテナのゲインテーブルのプロット  
スキャン間隔で解があることを確認

# self-calibration: amp.&phase

- 位相と振幅の補正テーブルを合わせて元ビジビリティデータに適用
  - gaintable=['self\_2.pcal','self\_ap.cal']
- テーブル適用後、最後のCLEANをする



# 分子輝線の解析について

- 連續波のself-calibrationで作成した位相/振幅補正テーブルを、分子輝線用ビジビリティデータに適用する
  - CASAタスク: applycal
    - vis='TWHydra\_corrected.ms.contsub'
    - gaintable=['self\_2.pcal','self\_ap.cal']
  - もちろん、SNが高く構造がシンプルであれば、分子輝線データを使ってself-calibrationをしてよい
- チャンネルでaveragingが出来ないので、データが膨大になる。少なくとも分子輝線毎(spw毎)にデータを分けた方が良い。  
=> CASAタスク: splitを使用
  - spw='0' or '2'としてspwで切り分ける

# 分子輝線の解析について

- CASAタスク: clean
  - 連続波解析と異なるのはmodeパラメータの部分
  - mode: 'velocity' or 'frequency' or 'channel'
  - start: イメージを作る際の1チャンネル目の値。modeで与えた変数の単位で入力  
ex) start='-4km/s'
  - width: チャンネル間の幅。modeで与えた変数の単位で入力  
ex) width='0.1km/s'
  - nchan: イメージを作る際の全チャンネル数。単位付きはなくチャンネルの数  
ex) nchan=118
  - restfreq: 着目する輝線の静止周波数。速度変換等で使用。  
ex) restfreq='345.79599GHz'
  - outframe: 速度のフレーム指定。 ex) outframe='LSRK'
- チャンネル毎にマスクをかけてCLEANをする必要がある。  
時間ががあればこれもinteractiveモードでやってみる

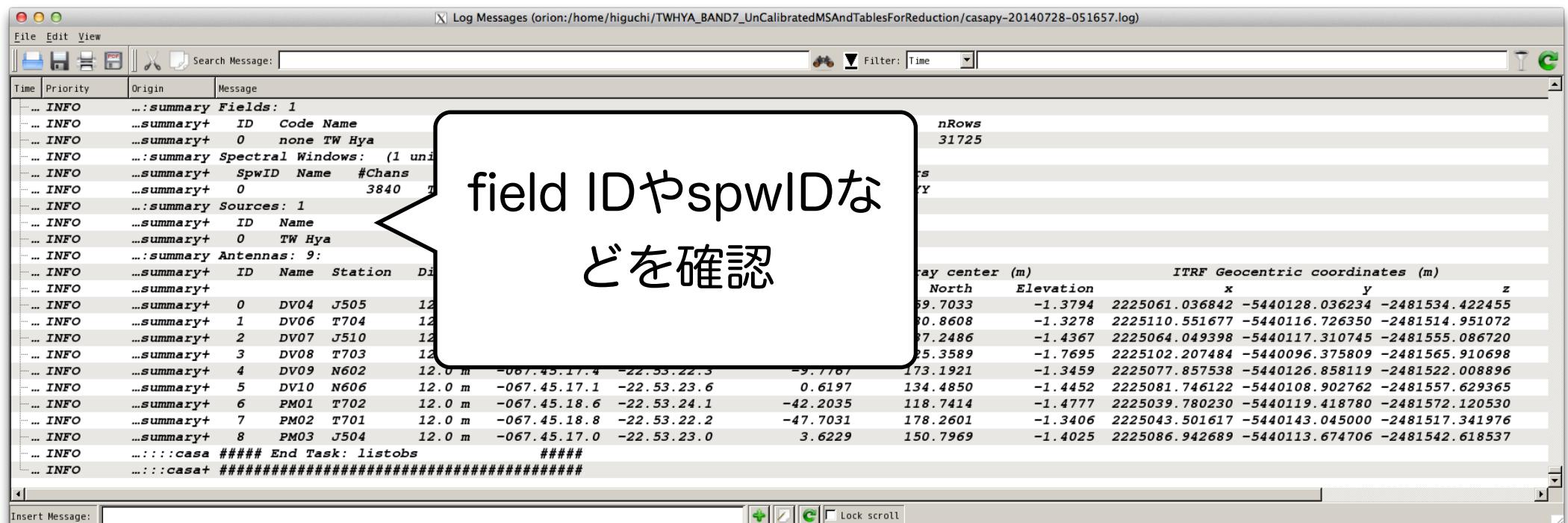
# TWHya CO(3-2): スペクトルチェック

CASA> listobs('TWHydra\_CO3\_2.ms')

CASA>plotms

もしくはターミナルでcasaplotms

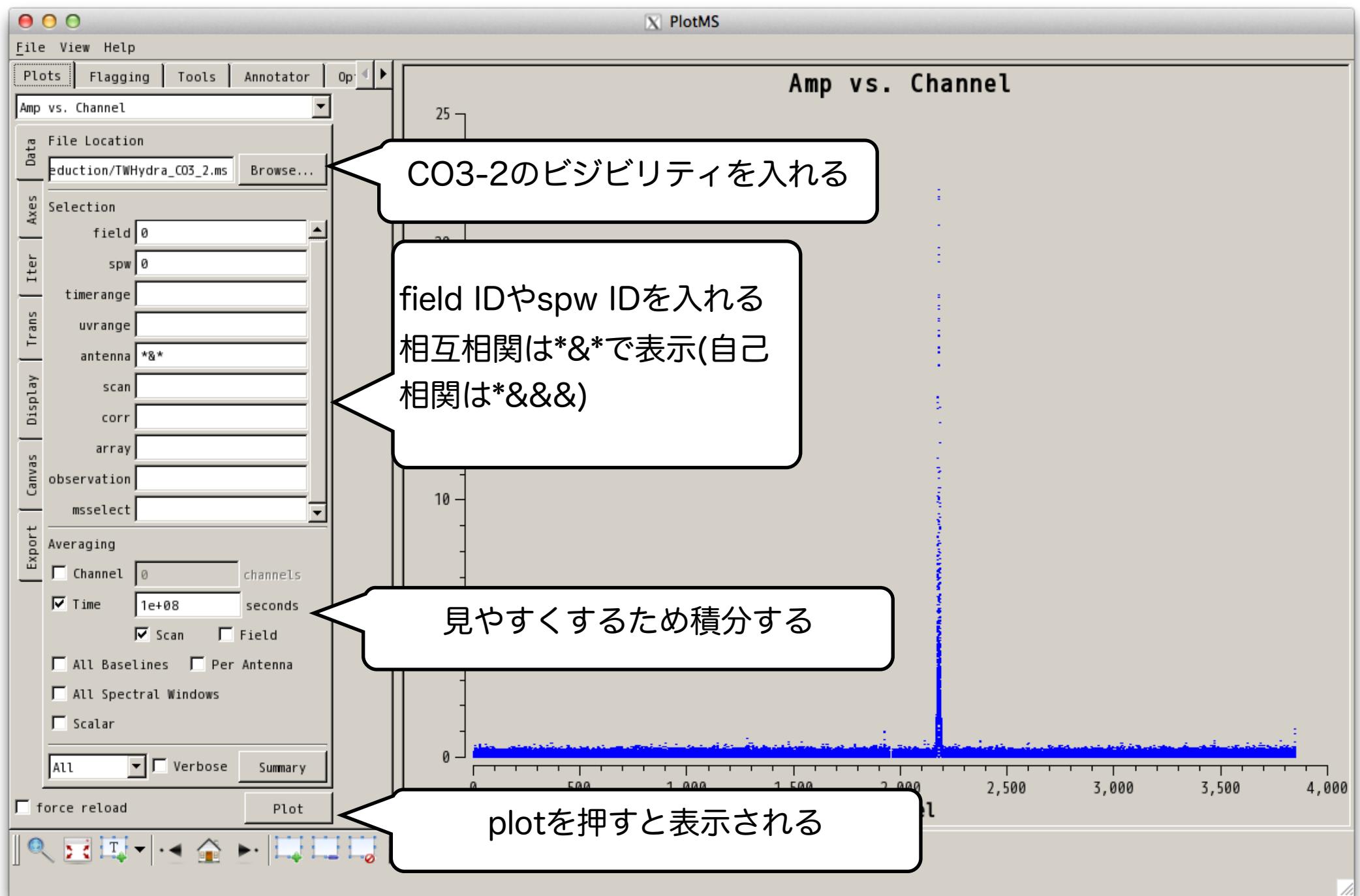
field IDやspwIDなど  
を確認



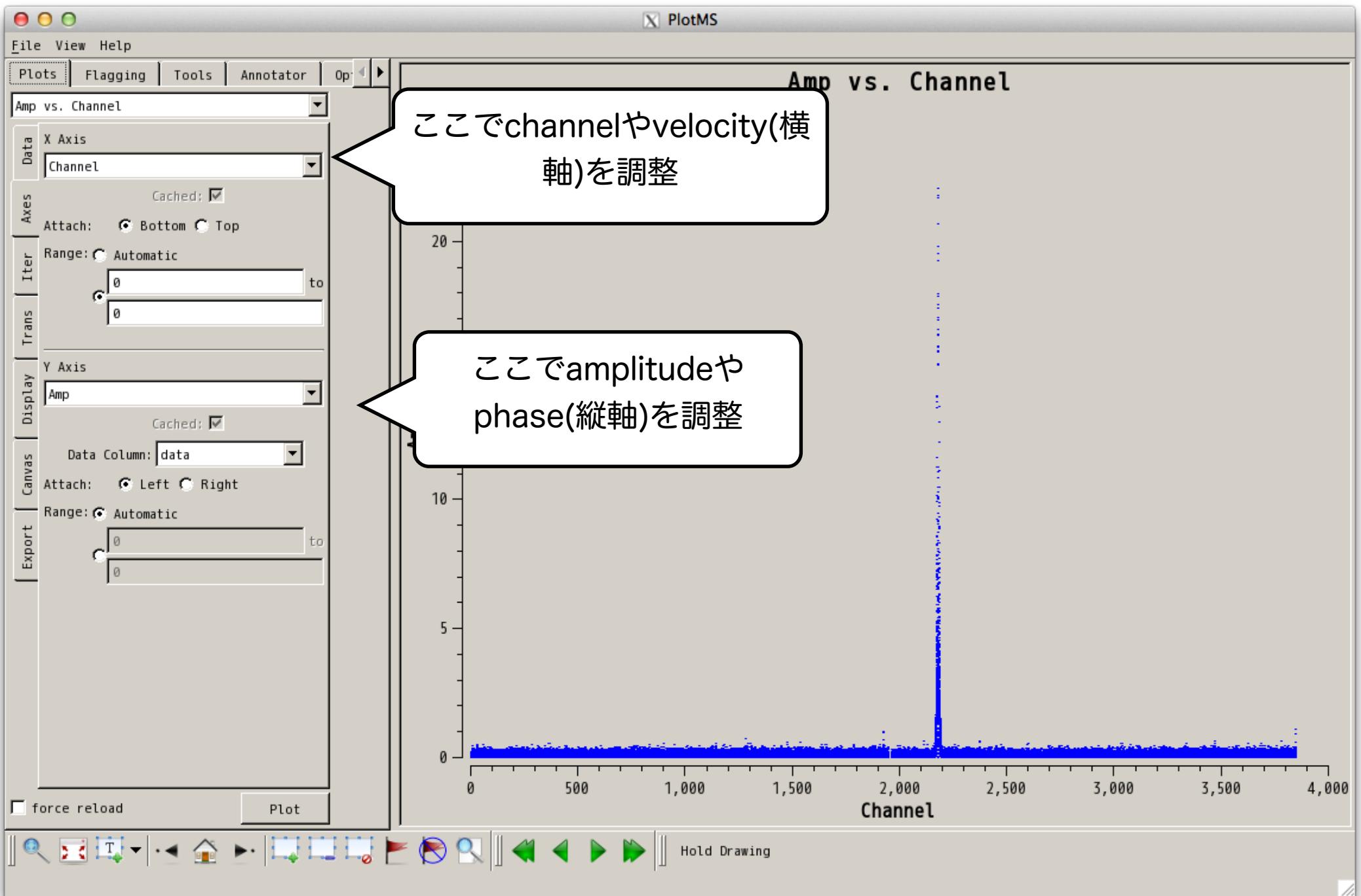
The screenshot shows the CASA Log Messages window with the title "Log Messages (orion:/home/higuchi/TWHYA\_BAND7\_UnCalibratedMSAndTablesForReduction/casapy-20140728-051657.log)". The window has a toolbar at the top with icons for File, Edit, View, and various search/filter options. The main area displays log messages in a table format with columns for Time, Priority, Origin, and Message. A callout box points from the text "field IDやspwIDなど  
を確認" to the "Message" column of the log table. The log output includes details about fields, sources, antennas, and spw information. A large portion of the right side of the window is occupied by a scrollable table of ITRF Geocentric coordinates (m) for various天体 (Celestial Objects). The table has columns for object name, Right Ascension (RA), Declination (Dec), and ITRF Geocentric coordinates (x, y, z).

Object	RA	Dec	x	y	z
1	12h 0m 0s	-067d 45m 17s	22.53.22.3	-1.3794	2225061.036842
2	12h 0m 0s	-067d 45m 17s	22.53.23.6	-1.3278	2225110.551677
3	12h 0m 0s	-067d 45m 18s	22.53.24.1	-1.4367	2225064.049398
4	12h 0m 0s	-067d 45m 18s	22.53.22.2	-1.7695	2225102.207484
5	12h 0m 0s	-067d 45m 17s	22.53.23.0	-1.3459	2225077.857538
6	12h 0m 0s	-067d 45m 17s	22.53.22.3	-1.4452	2225081.746122
7	12h 0m 0s	-067d 45m 17s	22.53.22.3	-1.4777	2225039.780230
8	12h 0m 0s	-067d 45m 17s	22.53.22.3	-1.3406	2225043.501617
9	12h 0m 0s	-067d 45m 17s	22.53.22.3	-1.4025	2225086.942689

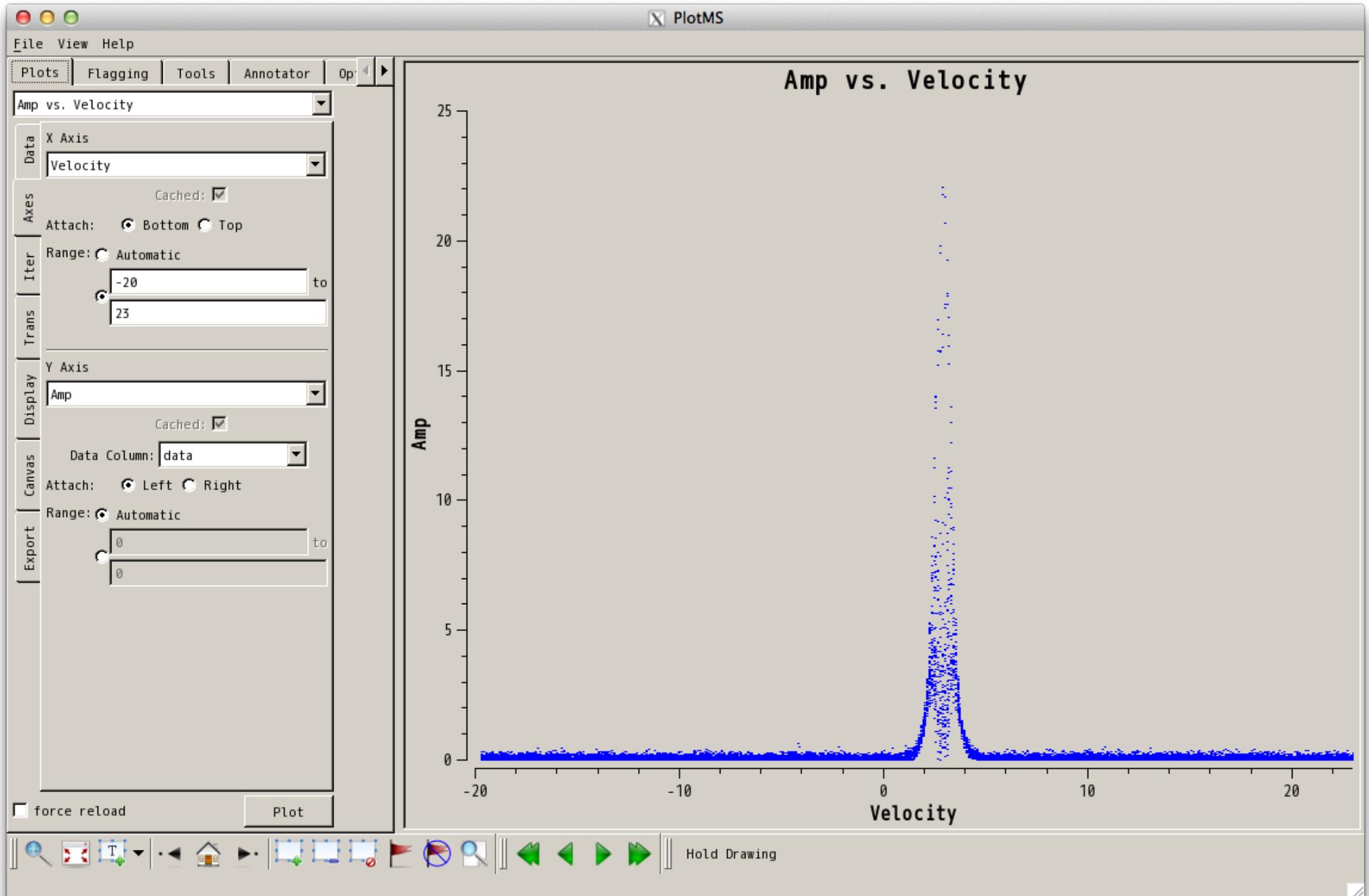
# TW Hya CO(3-2): スペクトルチェック



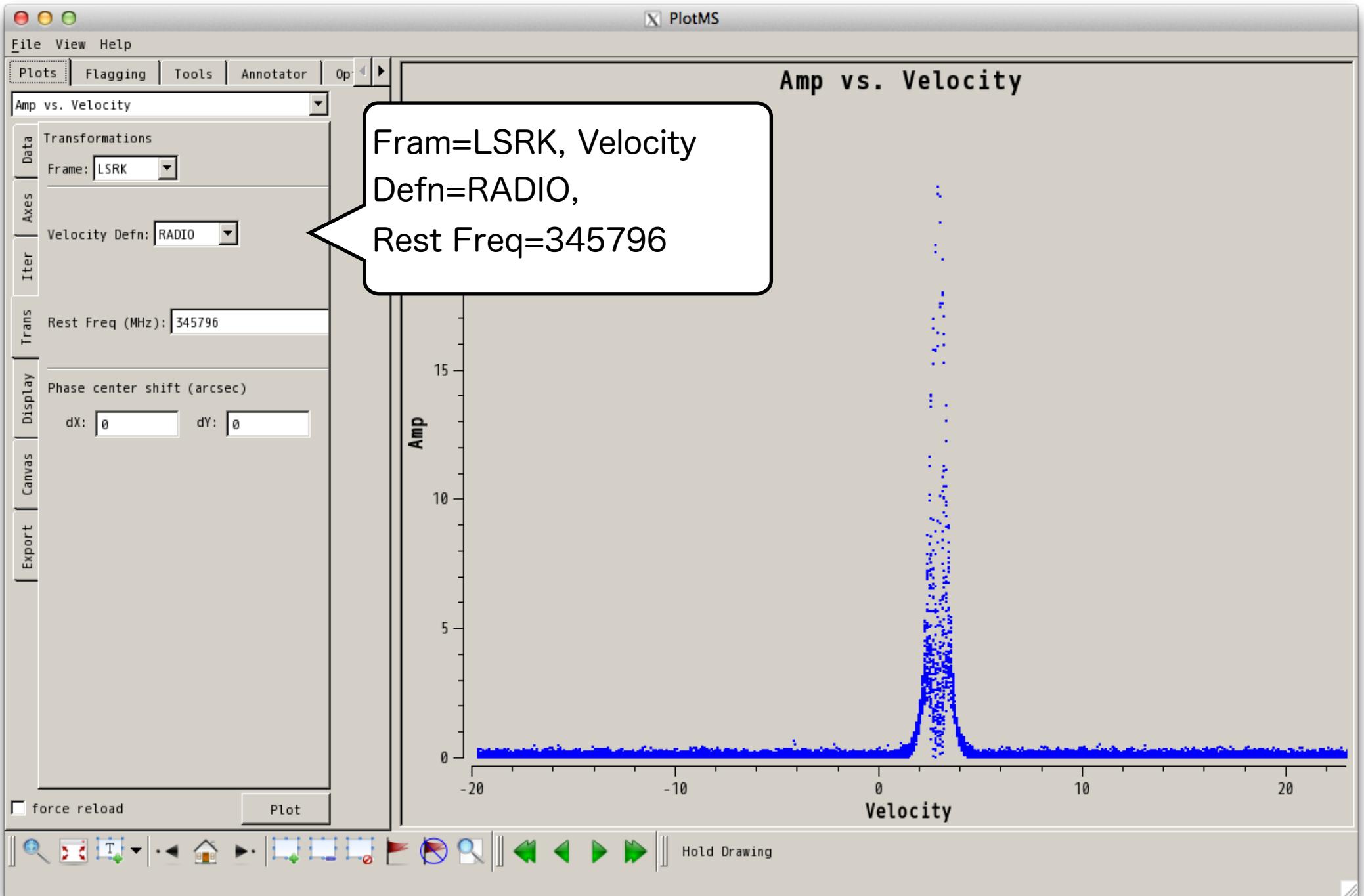
# TW Hya CO(3-2): スペクトルチェック



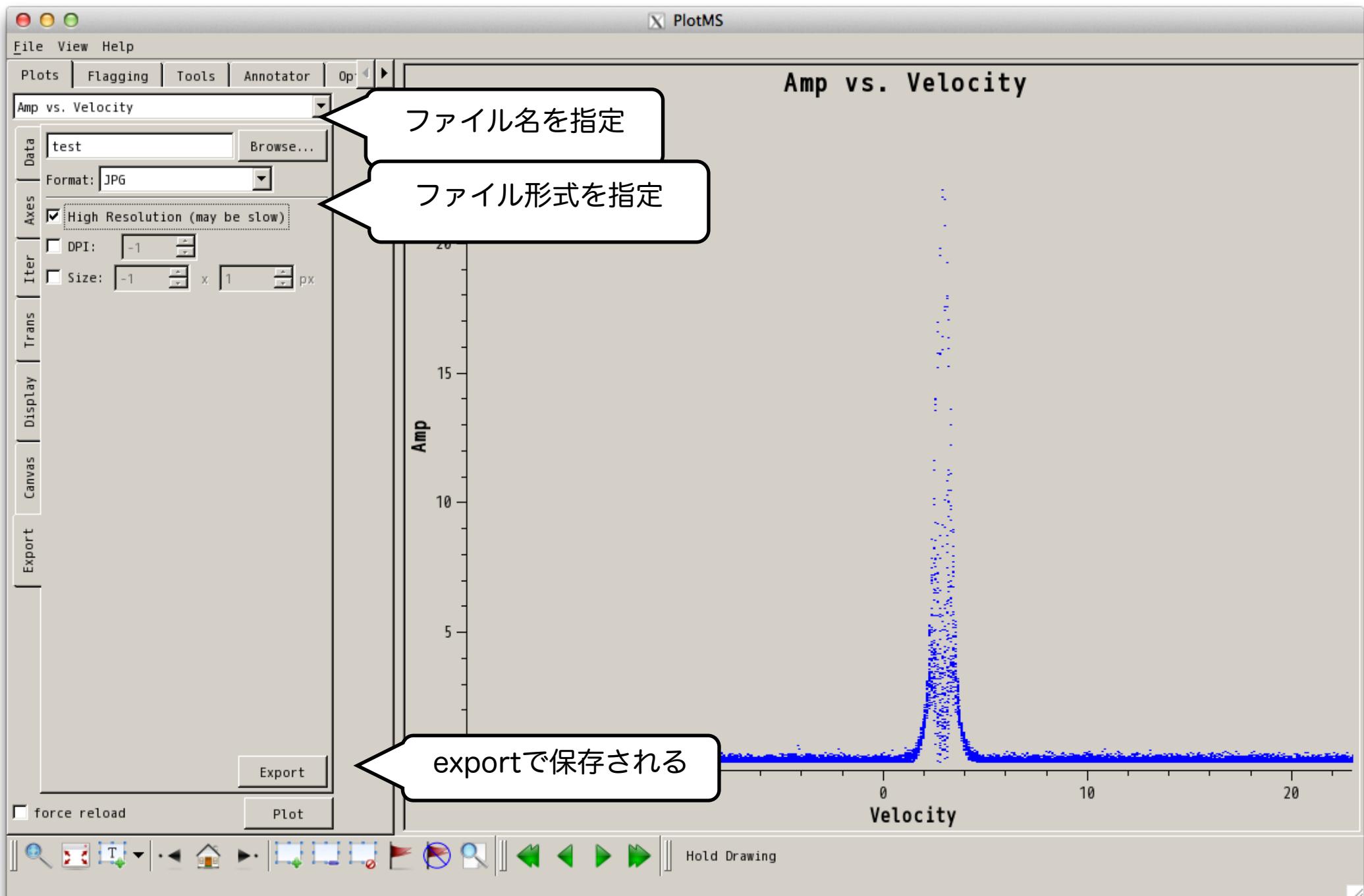
# TW Hya CO(3-2): スペクトルチェック



# TW Hya CO(3-2): スペクトルチェック



# TW Hya CO(3-2): スペクトルチェック



# モーメントマップの作り方

- CASAタスク - immoments
  - moments: モーメントマップの次数
  - chans: 計算に使用するチャンネル(≠速度、周波数)
  - outfile: 出力するファイル名。空欄だと入力ファイル名+適切なsuffix
  - axis: モーメントを計算する軸。デフォルトはspectral軸。
- 1,2次モーメントマップはノイズレベルでマスクをかけた方が良い
  - $5\sigma$ 以上が目安
  - includepix: 計算に使用する範囲を入力する  
ex) includepix=[0.05, 99999] ( $1\sigma=0.01$ )
  - excludepix: 計算に使用しない範囲を入力する  
ex) excludepix=[-99999,0.05] ( $1\sigma=0.01$ )

# FITSによる保存

- CASAタスク: `exportfits`
  - イメージデータをFITSファイルに出力
  - `velocity`: イメージファイルが3Dの場合、周波数軸を速度に直すかどうか。デフォルトは`False`。
  - ex)  
`exportfits(imagename='TWHydra_CO3_2line.image',fitsimage='TWHydra_CO3_2line.image.fits',velocity=True)`
- CASAタスク: `uvexportfits`
  - ビジビリティデータをFITSファイルに出力